"Express Mail" mailing label number:

EL487742543US

E-SERVICE TO MANAGE CONTACT INFORMATION WITH PRIVACY LEVELS

James G. Douvikas
Terry R. Sheehy
Chris W. T. McKay

CROSS-REFERENCE TO MICROFICHE APPENDIX

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the patent and trademark office patent file or records, but otherwise reserves all copyright rights whatsoever.

BACKGROUND OF THE INVENTION

Field of the Invention

The present disclosure relates to electronic commerce, more specifically electronic storage and retrieval of information.

Description of the Related Art

The ordinary paper business card has become ubiquitous worldwide. Social rituals have even developed concerning the exchange and scrutiny of these small slips of cardboard. By some estimates, billions of business cards change hands every day, yet the cards themselves have numerous shortfalls. Each business card contains only static information on the cardholder, i.e., the person for whom the card was printed and whose name is on the card. Business cards must be reprinted every time any cardholder information (such as a phone number, electronic mail [email] address, or title) changes. Business cards consume not inconsequential amounts of space, yet lack

an inherent card-to-card organization. Thus, it can be difficult to retrieve information from a stack of cards, especially if that stack numbers in the hundreds or thousands of cards.

Privacy of information is also a growing issue among modern business people. By definition, the information on a card is public, yet other information (such as a mobile or home phone number) is necessarily shared with some acquaintances. In such situations, the cardholder or recipient must fumble for a pen and the additional data must be dictated and captured.

Dynamic access to the cardholder by others is not addressed by the prior art business card, as it only shows static location information as of the last printing of the card. Thus, if a business person is based in Huntington, New York but happens to be traveling to San Jose, California, that person's business card will not reflect the California address or phone numbers.

Electronic means of capturing and storing conventional business card data are currently known. Examples of this technology include card scanners, personal digital assistant (PDA) devices and related software, electronic address books, commercial email programs such as Microsoft® Outlook having their own electronic address books, "smart phones" or PDA/wireless communication device hybrids, Internet (also referred to as the World Wide Web, or simply "Web") based contact organizers, and the like. This technology all suffers from the same limitation in that it generally lacks multi-level privacy for users and cardholders, it cannot help locate the cardholder, it (generally) lacks the ability to seamlessly export to or import from other database systems, and (with the possible exception of some prior art Web-based contact organizers) it lacks centralized control and universal access.

What is needed is a widely-accessible electronic service and method for organizing contact information entered by cardholders, including but not limited to all of the information on a standard business card. This service must provide for the ability to export data to standard databases. Privacy of information should be configurable at an information record and field level by the cardholder so that access to some records and some fields in all records can be denied to certain people while

access to other records and fields is still allowed. A location feature to allow service users to determine how to best reach a listed cardholder at a given time is also desirable. A dynamic electronic link, such as the well-known Internet hyperlink, is also needed to connect the recipients of email from a cardholder to the service.

SUMMARY

In one embodiment of the present invention, an electronic business card (EBC) access and organization system operates from a Web-based computer system that includes a database and software for managing access, data privacy, and dynamic updates. The cardholder database, i.e., the database containing records of each registered cardholder (or "Member" of the EBC system), is accessible from any Web browser connected to the Internet. Examples of such common Web browsers are Microsoft's Internet Explorer and Netscape® Navigator®. In an alternate embodiment, the EBC system may be installed behind a conventional network "firewall" security device and thus made accessible only to browsers connected to and authorized to use the intranet defined by and behind the firewall.

Access to and delivery of contact information in the EBC system is not limited to a Web browser interface as commonly known today. Some embodiments of the present invention provide multi-mode access interfaces, including but not limited to interfaces using voice-controlled and conventional wireless PDA and/or cell phones, two-way pagers, and wireless access protocol (WAP)-enabled devices. Further embodiments of the present invention provide data delivery interface embodiments using, for example, the common alphanumeric pager, wireless markup language (WML), or voice delivery (e.g., audio playback) systems commonly used in the art.

Users (those desiring access to one or more cardholder records) are permitted to search for cardholder information. Access to individual records is controlled at both the record level and the field level. Users having certain permissions (set by the cardholder) are permitted to read a defined group of records, though not necessarily all fields in each record. Thus, a cardholder may make her business information available to all users (or all users in a defined group or groups, such as "Aerospace Engineers"

or "Family"), but keep certain information, such as her cellular phone number, private to all but a few individuals. Access control is implemented, in some embodiments of the present invention, with multiple privacy levels for each field, in addition to the well-known "public" and "private" levels. In these embodiments, the cardholder can specify the degree of privacy associated for each field. For example, a cellular phone number may be designated as semi-private and thus available to only those defined users granted "semi-private" access to the cardholder's data. In an alternate embodiment, more than three privacy levels are defined, allowing even more degrees of privacy and finer-grained access control.

BRIEF DESCRIPTION OF THE DRAWINGS

The present disclosure may be better understood and its numerous features and advantages made apparent to those skilled in the art by referencing the accompanying drawings.

Figure 1 is a high-level schematic of the hardware platform, according to one embodiment of the present invention.

Figure 2 is a flowchart of the "Search" process, according to one embodiment of the present invention.

Figure 3A is a screen shot of the Member Login display, according to one embodiment of the present invention.

Figure 3B is a screen shot of the Search display, according to one embodiment of the present invention.

Figure 4 is a screen shot of the Card Display screen, according to one embodiment of the present invention.

Figure 5 is a flowchart of the "Become New Member" process, according to one embodiment of the present invention.

Figure 6 is a screen shot of the Terms & Conditions display, according to one embodiment of the present invention.

Figure 7 is a screen shot of the New User data entry display, according to one embodiment of the present invention.

Figure 8 is a screen shot of the Registration Confirmation display, according to one embodiment of the present invention.

Figure 9 is a screen shot of the Welcome display, according to one embodiment of the present invention.

Figure 10 is a screen shot of the My ecardfile display, according to one embodiment of the present invention.

Figure 11 is a flowchart of the Export process, according to one embodiment of the present invention.

Figure 12 is a screen shot of the File Maintenance display, according to one embodiment of the present invention.

Figure 13 is a screen shot of the Export display, according to one embodiment of the present invention.

Figure 14 is a flowchart of the "Where Am I?" process, according to one embodiment of the present invention.

Figure 15 is a screen shot of the "Where Am I?" display, according to one embodiment of the present invention.

Figure 16 is a flowchart of the signature hyperlink export process, according to one embodiment of the present invention.

Figure 17 is a screen shot signature hyperlink export display, according to one embodiment of the present invention.

Figure 18 is a function block diagram of the Boomerang software application, according to one embodiment of the present invention.

Figure 19A is a schematic map of some of the database relationships, according to one embodiment of the present invention.

Figure 19B is a schematic map of some of the database relationships, according to one embodiment of the present invention.

Figure 19C is a schematic map of some of the database relationships, according to one embodiment of the present invention.

The use of the same reference symbols in different drawings indicates similar or identical items.

DETAILED DESCRIPTION

Introduction

The electronic business card (EBC) access and organization system consists of a hardware complex providing the physical interface to the Internet, firewall security, web server functionality, data storage, and system redundancy protection. The hardware is controlled and operated by computer instructions (i.e., software) in various forms, including but not limited to microcode, firmware, assembly and other high-level language modules. The EBC system is the integration of the hardware and software elements to perform the functions and provide the features noted in the Summary above.

Hardware Platform

In one embodiment of the present invention, the EBC access and organization system (also referred to as ecardfile™, the Hewlett-Packard® product embodying a certain aspect of the present invention) is run on 4 N-class Hewlett-Packard 9000 computers 110, as shown in Figure 1 . These computers are configured, in one embodiment, with 4 Gigabytes (GB) of memory and 4 processors, running the HP-UX® operating system version 11.0. Other memory/processor/operating system configurations are also possible. There are two front-end web servers (110A and 110B) talking to a database server (110C) that has access to one or more shared automatic redundant arrays of inexpensive disk drives (RAIDs) 120, each RAID having 64 GB of memory. Warm fail-over server (110D), which uses in one embodiment MC-Service Guard™, a Hewlett-Packard product, protects database server 110C.

Although an N-class Hewlett-Packard 9000 is described, those skilled in the art will realize that servers other than the N-class Hewlett-Packard 9000 can be used. Accordingly, the invention is not limited to any particular type or brand of server.

- *Web Servers*

The two front-end web servers 110A and 110B are served by a two Cisco Systems®, Inc. Catalyst™ 5505 switches 130A and 130B, which are served by two Cisco Local Director™ units 140A and 140B, which in turn are server by two Cisco PIX® Firewall units 150A and 150B. The firewall units 150 connect through an internet service provider (ISP) local area network (LAN) 160 to the Internet 170.

The web servers are running, in one embodiment of the present invention, the Stronghold® Apache web server operating program, available from C2Net, Inc. This is a 128 bit secured web server. Other commercially-available server operating programs are also useable.

The Local Director units 140A and 140B load balance the incoming requests to the two web servers 110A and 110B by switching packets in switches 130A and 130B.

The web content (resident in web servers 110A and 110B) is dynamically built with the aid of server-side Java™ applications known as servlets. The JRun™ servlet engine (in one embodiment of the present invention, version 2.3, build 145) executes these servlets, with the aid of a dynamic loaded module within the Apache web server operating program. The servlets are written to conform to the Java Servlet Development Kit API v2.1, available from Sun Microsystems, in order to properly interface with the Apache web server software.

The biggest advantage in using servlets as opposed to other web development tools is performance. A single Java virtual machine (in one embodiment of the present invention, the HP-UX Java Virtual Machine version 1.1.8.1) runs on the server and the servlet is loaded once when it is called. It is not loaded again until the servlet changes, and a modified servlet can be re-loaded without restarting the server. The servlet stays resident in memory and is very fast. Static or persistent information can

be shared across multiple invocations of the servlet, allowing the sharing of information between multiple users. For instance, a single database connection can be use by multiple browser requests.

- *Database Server*

The database servers 110C and 110D run, in one embodiment of the present invention, Informix® Dynamic Server® Version 7.31.UC4. Other database server software packages and versions are also useable. Access to the database is via the industry standard JDBC applications programming interface (API) and Informix' JDBC drivers (in one embodiment of the present invention, driver version 1.40.JC2). The use of JDBC ensures scalability and database and platform independence.

User Interaction

A user's interaction with the ecardfile EBC system, according to one embodiment of the present invention, is illustrated in Figure 2. This flowchart shows a typical use of the EBC system to perform a search on cardholder data. Note that, in some embodiments, the user need not be a registered Member of the ecardfile system. In an alternate embodiment, only registered Members can use the system.

Interaction begins at step 200. The user starts a conventional Web browser, such as Internet Explorer or Netscape Navigator, 210, and enters the ecardfile Uniform Resource Locator (URL) 215. This URL is the Internet address of the EBC system hardware described above and is defined, disseminated, and maintained through means well-known in the art. Upon receipt and processing of the URL by Internet 170 (generally speaking), the user's browser is redirected to a secure web site 218 by conventional techniques common in the art.

Once connected, ecardfile returns the web browser codes (e.g., HTML) to the user to display the Member Login screen, shown in one embodiment in Figure 3A. The Member Login screen display consists of window 310 containing the ecardfile login graphic 320, copyright notice 330 and hyperlinks 340 and 345, to a terms of use page and a privacy statement, respectively.

In step 230 of Figure 2, the user selects the "Go" button (350) associated with the "View Cards" command in login graphic 320. This selection brings up search screen 360 (Figure 3B) in window 310. In one embodiment of the present invention, the user is given the choice 232 of searching by cardholder name 234 or ecardfile ID number 236, a multi-digit number or multi-character alphanumeric value representing a cardholder. If the user selects lookup by name, the user is given the further choice 238 of searching by similar sounding names using, for example, the SoundEx software toolbox. In the latter case, the user checks check box 365 to conduct a sounds-like search.

The search begins 250 when the user selects the "Go" button 370 or 375 corresponding to the type of search desired in step 240. The EBC system returns results 260 in the Results Screen 410 shown in Figure 4. This screen replaces search screen 360 in window 310.

A user may chose, at step 220 of Figure 2, to become a Member of the ecardfile user community, rather than performing a search. In this case, the interaction follows the flowchart of Figure 5 instead. The process begins in step 501. The user starts a Web browser 210 and visits the ecardfile login screen 320 as before (steps 215, 218, and 220). Here, however, the user selects the "Become New Member" button 370 (Figure 3A) in step 510.

This selection brings up Terms screen 610 (Figure 6) in window 310 (step 515). The user is then given choice 520 to go back to login screen 320 (which ends the "Become New Member" process, step 599) or to continue the process to step 525. Choice 520 is implemented, in one embodiment of the present invention, using Back button 620 and Continue button 630.

The selection of Continue button 630 brings up, in step 525, New User data entry screen 710 in window 310 (shown in Figures 7A and 7B). The user selects an ecardfile ID and password and fills out the personal data using conventional data entry means for any of the well-known access devices and systems discussed above, such as using a keyboard or a voice-controlled (audio) prompt and response scheme. Privacy levels 720 are also set by the user for each field in the user (soon-to-be Member and

Cardholder) data record. In one embodiment, depicted in Figures 7A and 7B, three privacy levels are possible: Private, denoted by the locked padlock icon 722; Semi-Private, denoted by the partially-locked icon 724; and Public, denoted by the open lock icon 726. Public and Private privacy levels represent the familiar "all access" and "no access" privacy settings known in the art. Semi-Private privacy represents a level of access granted only to certain members of the public, such as a predefined group of users designated by the Member or by a EBC system administrator. Non-designated users do not have access to Semi-Private fields.

Although a three-level (Private, Semi-Private, and Public) privacy scheme is described, those skilled in the art will realize that privacy (or security) schemes implemented in more than three levels can be used. For example, a variety of different user groups can be defined with exclusive Semi-Private access given to some groups over others on a field-by-field basis. Accordingly, the invention is not limited to any particular number of privacy levels.

Once all requested information has been entered (or left blank, where optional as defined by the EBC system), the user chooses to continue or not at step 535. If the user decides not to continue by pressing Cancel button 740 (Figure 7B) , the process ends at step 599. If, however, the user chooses Okay button 730, the system displays a Registration Confirmation page 810 (Figure 8) in window 310. Here the user is given the choice 545 to accept the terms of use of the EBC system (via Accept button 820) or to go back (via Back button 830) to step 530 to edit personal data or enter additional data.

If the user accepts the terms of use, and email message is automatically dispatched 550 by the EBC system to the new Member's designated authorizing email address (field 750 in Figure 7B). A welcome screen 910 (Figure 9) is then displayed to the new Member, step 560, in window 310.

Meanwhile, the user (now a Member) receives an authorization email message by conventional means. In one embodiment of the present invention, such an email message is as shown in Table 1 below. This email authorization method provides a measure of additional security by ensuring that each Member is associated with a valid

email address. The email address is also used to verify a user identity in case of a lost password: on the user's request, the password will be sent to the authorized email address only.

When the user selects the embedded hyperlink (in this example, the string beginning *http://ecardfile.com...*) in step 570, the EBC system directs the Member (in step 580) to home page 1010 in window 310 (Figure 10) . In some embodiments (not shown), home page 1010 is personalized with Member data, such as the Member's name.

Table 1

Welcome! You have been added as a new Member of ecardfile.com. To activate your password, please click on the Web address below: *http://ecardfile.com/search?op=Confirm&eCardId=terry&createId=0Av4Jj6nWZwA4* You can also type the above Web address into your Web browser. Once you have connected to the ecardfile.com site through the above address, your password activation is complete, and you can begin to enjoy the ease and convenience of having your business contact information on the Web. If you do not connect to ecardfile.com through the above address within two weeks, your card will be deactivated. We look forward to having you as an ecardfile.com Member! webmaster@ecardfile.com

Export Process

Figure 11 shows the process whereby a user is able to export card data to a file. As above, the user must first login to ecardfile.com. At step 1610, the user selects file maintenance button 394. A user file maintenance screen (Figure 12) is then displayed, step 1620. The user selects a card to export, step 1630, from the list of cards added by the user to his/her private list. These cards represent other users to whom the user has granted extra privileges. Non-member users do not have export privileges. Of course, only the information to which the user has been granted access is exported.

The user next selects the export button 1710 (Figure 12) in step 1640, which brings up an export options screen, shown in Figure 13. The user selects the desired output format, step 1660, and is presented with a conventional "Save As" dialog, step 1670. The user enters a name of the file into which the card export will be saved, step 1680 and selects the Save button, step 1685. If Save is selected, the data will be saved by conventional means in step 1687. If not, as when the user selects "Cancel" instead, the process drops to the end state, step 1690, and returns to waiting for user input.

The exported card is formatted into a pre-defined data file structure readable by one or more conventional and commercially-available contact management programs. In some embodiments, custom export file formats may also be defined by the user to provide even wider connectivity and cross-platform utility.

"Where Am I?" Contact Location Tracking

The process of setting up a temporary location pointer for a specific Cardholder/Member begins in step 1110 of Figure 14. As before, the Member sets his or her browser to the appropriate EBC system URL and connects to the system, steps 210, 215, 218, and 220. The Member then logs in, step 1120, and is displayed home page 1010 (Figure 10) in step 1130.

The Member then selects an icon or button denoting the function "Edit My Card." In one embodiment of the present invention, this function is iconified in button 395. In response, the EBC system displays (step 1140) a user information screen (not shown) in window 310. The Member there selects a button denoting the function "Where Am I?" in step 1150.

At this point (step 1160), the EBC system displays "Where Am I?" screen 1210 (Figure 15) in window 310 thereby prompting the Member for a phone number and additional details of the Member's location, step 1165. This information also includes an expiration date, i.e., a date beyond which the "Where Am I?" data is no longer valid.

To exit the "Where Am I?" information dialog, the Member clicks Okay button 1220 (step 1167), to save the "Where Am I?" data, or the Back button 1230 to

cancel "Where Am I?" data input. The process returns to a display of the user information screen, step 1140, and stops, step 1199.

Signature Hyperlinking

The process of exporting a signature hyperlink, shown in Figure 16, is almost the same as that of exporting a card, except that the user selects the My Card button 395 rather than the file maintenance button 394. The options screen, shown in Figure 17, is presented at step 1820. The output of the process, step 1830, is a file containing either a hyperlink or a conventional vCard file, as selected from the options display of Figure 17.

Either the signature hyperlink or the vCard (which can also contain a hyperlink) can then be used by conventional email programs. Electronic mail sent by the cardholder is automatically formatted to contain a signature hypertext link, according to the well-known hypertext markup language (HTML) standard or any of its common variants, directing recipients of the email to the electronic business card access and organization system. This hyperlink enables the recipient of the email to rapidly access the EBC system to locate the cardholder and/or obtain additional information. In effect, receipt of an email containing the hyperlink enables the recipient to easily become a user. In some embodiments, the signature hyperlink is part of the vCard feature known and implemented in common email programs such as Microsoft Outlook and Netscape Communicator®. In an alternate embodiment, the signature hyperlink is implemented using the well-known email signature block feature.

ecardfile Help Screens

The following topics are the subject of individual help screens, available to any EBC system user or Member by pushing (in some embodiments of the present invention) the Question Mark ("?") button 390, shown in, e.g., Figure 3A. The contents of these help screens are reproduced below as an aid to understanding the EBC system.

- Become a Member

- Set up your card
- Set up "Where Am I?"
- Add others' cards to your ecardfile
- Exchange cards with others
- Set/change privacy levels
- Export cards to your address book
- Set up your email signature
- Contact ecardfile.com support

- *Become a Member*

As an ecardfile Member, you set up your own Card Profile and establish your own unique Card ID and Password. Then, whenever you log in, you are located at your personal ecardfile and can view cards from other Members and add them into your ecardfile.

Let's walk through the process of becoming a Member.

1. From the Member login screen, click the Become a Member button.
2. Fill in your Card Profile: the profile contains all of your contact information and can be updated as needed. See the help menu topic "Set Up Your Card" for more information.

After your membership is confirmed, you can log in to ecardfile.com using your Card ID and password. After log in, you are brought to your personal ecardfile area. Here is where you can store other Member cards and perform functions such as adding, deleting, changing the privacy level access to your Card that you have given to other Members, and exporting a card to your address book.

It's a good idea to keep your "Where Am I?" information current. To access it, click on the My Card Profile icon and scroll to the bottom of the screen.

The more people you know who join ecardfile.com, the easier it is to use this convenient way to access business information. To help get the word out, please download your personal email signature or vCard (accessible from the export button at the bottom of your Card Profile) and attach it to all your emails. When people receive email from you, they can click on your signature link and go directly to your Card Profile in ecardfile.com. From there, they can also choose to become ecardfile Members, if they are not Members already.

- *Set Up Your Card*

When filling in your Card Profile, please keep a few concepts in mind: When selecting your Card ID, use up to 40 alphanumeric characters. Because you will be giving out this ID to your business and personal contacts, make your ID simple and easy to remember; it's use is similar to that of your email username. Although ecardfile.com also enables you to be looked up through a first name/last name search, it will usually be much faster for people to look you up by your Card ID.

When selecting your password, use up to 10 alphanumeric characters. Make your password something easy for you to remember and hard for others to guess.

As you are entering information into your Card Profile, please keep in mind that ecardfile.com gives you three levels of privacy for each field:

Level 1 - Public. Information at this level will be displayed to anyone who looks up your card. This could be anyone viewing cards from the World Wide Web, whether you know them or not.

Level 2 - Semi-Private. Information at this level will displayed only to other ecardfile Members who are in your personal ecardfile and who have been designated to receive your semi-private information.

Level 3 - Private. Information at this level will displayed only to other ecardfile Members who are in your personal ecardfile and who have been designated to receive your private information.

All field information is set to private when you first fill out a Card Profile. Be sure to select other privacy levels for fields that are either semi-private or public.

The Email Auth field is used only by ecardfile.com for verification purposes. It is never displayed on your Card. You must enter a current email address in the Email Auth field. Once you complete the Card Profile and click "OK," ecardfile sends an email to this address and waits for your reply before authorizing your membership and enabling you to log in. This authorization process has been designed to protect your privacy and identity.

- *Add others' cards to your ecardfile*

From your personal ecardfile screen, use the Look Up fields to view the card of the Member you want to add. When the card is displayed, press the Add icon.

If you would like to give this Member access to your semi-private or private ecardfile information, be sure to change the privacy level displayed next to the Member's name. See the help topic "Set/change privacy levels" for more information.

- *Exchange cards with others*

In order to protect your privacy, ecardfile.com offers several ways you can exchange cards with others.

- Anyone, whether they know you or not, can look you up by name and see the information designated as "public" in your Card Profile. Note: for this reason, you may decide not to have your email addresses be part of your public information.
- Casual or new acquaintances can look you up by name search or by Card ID and see the information designated as "public" in your Card Profile.
- Members can look you up by name search or Card ID and see the public, semi-private or private information you have specifically designated for them.

For example, let's say a new person, Hans, has joined your project team; he works out of your company's Munich office, and you are in Los Angeles. Hans is not yet a Member of ecardfile.com.

1. At the initial team conference call, you give Hans your Card ID and tell him that's where all your contact information is.
2. Hans accesses ecardfile.com and becomes a Member. He looks up your Card and adds it to his personal ecardfile. He wants you to have his semi-private information so he marks your card with the semi-private access key.
3. You look up Hans's card and add it to your personal ecardfile. You then mark Hans's card with the semi-private access key so that he can see more detailed information about you than what appears on your public card.
4. Over the next 6 months, Hans changes office locations and gets a new phone number; your fax number changes, and the Post Office gives your part of town a new zip code. Thanks to ecardfile.com, your contact information is always current.

- *Set Up "Where Am I?"*

From your Card Profile, scroll to the bottom of the screen and click on "Where Am I?" You'll have the option to input a current phone number, details about your whereabouts, and an expiration date. Note that you can specify different privacy levels for the phone number, details and expiration fields, so you might want your phone number to be public, while the details of where you are remain private. To see another Member's "Where Am I?" information, select her card from your personal ecardfile or look up her card. When the card is displayed, scroll to the bottom of the screen and click on "Where Am I?"

The expiration date uses the date at the ecardfile server's location, US Pacific Standard Time (PST). The expiration date is customizable to the ecardfile Member's own location.

- *Set up your email signature*

Use this function to download a signature file or a vCard from ecardfile.com to your email system. Access it by going to your Card Profile, scrolling to the bottom of the screen, and clicking on Export. Then follow the instructions on the screen to export to your particular email system.

A signature file has an HTML link to your Card; when downloaded, the signature file will embed the link into all of your email messages. When someone reads your message and wants to view your contact information, he just clicks on the HTML link and is immediately connected to your Card and your up-to-the-minute contact information.

A vCard is a file that holds your contact information in a standard format. Some email packages such as Microsoft Outlook and Netscape Communicator recognize this format and can treat it in a special way. Because it is not a live link, it may display old or inaccurate information, particularly if someone is reading an old email message from you.

If your email package, or more importantly the message recipient's email package, does not support HTML tags or vCards, you may cut and paste the HTML link displayed and attach it to your messages. The recipient just clicks or cut and pastes the HTML link into a browser and is immediately connected to your Card and your up-to-the-minute contact information.

- *Export cards to your address book*

Use this function to download cards from your personal ecardfile to your email address book.

1. Go to your personal ecardfile.
2. Select the card for export by placing a check mark next to it.
3. Click on the maintenance button.
4. Select export.

5. Select the format of your address book.

6. Follow the online instructions to export the card information.

- *Contact ecardfile.com support hyperlink*

Here, an email window automatically pops up if this link is selected, and ecardfile.com's technical support address is automatically inserted into the "To" field.

- *Get help on a specific screen, field or icon*

To get help on a specific screen or field, place the mouse arrow on the gray bar of the screen and press the "Help" key on the keyboard.

To get help on a specific icon, pass the mouse over the icon, and the icon title will display.

One of ordinary skill in the art will appreciate that many other methods of activating context sensitive help are known in the art; the present disclosure is intended to encompass all such well-known methods and is not limited to any single form.

- *Set/Change privacy levels*

ecardfile.com gives you two ways to control who sees what information about you: You can designate each field in your Card Profile with a specific privacy level that governs private, semi-private or public viewing of that field. Alternatively, you can specify which Members have access to which level of privacy about you. If a particular Member has access to your semi-private information, he or she will see all of the fields marked public or semi-private when viewing your card.

- *Designate Your Card Profile privacy:*

To change privacy levels on a field in your Card Profile (for example, to make your business email address, which had been public, semi-private) click on the My Card Profile icon and then click the appropriate new privacy button next to the email address field. A public field has the open padlock icon selected. A semi-private field

has the partly-open padlock icon selected. A private field as the closed padlock icon selected.

- *Specify Member privacy:*

Let's say you want to give your new manager access to your private information.

1. From your personal ecardfile, look up your manager's card and add it to your file.
2. Select the card for maintenance by placing a check mark next to it.
3. Click on the Maintenance button.
4. Select Edit Privacy.
5. Select the new privacy level you want your new manager to have about you.
6. Select Update to save the new privacy level.

Now, whenever your new manager accesses your Card, she will see all information you have designated as public, semi-private and private.

Software Implementation

In one embodiment of the present invention, the controlling software application providing some of the EBC system functionality is called Boomerang. The Boomerang application has five major components, shown schematically in Figure 18:

1. Session Manager daemon 1310, which maintains state information;
2. Login Servlet 1320, which handles user/Member authentication;
3. Search Servlet 1330, which handles most of the user/Member interface functionality;
4. JDBC Objects/Classes 1340, which implement the database functionality; and
5. HTML Template Engine, which handles conventional dynamic HTML processing. This component is typically implemented in Java and reads HTML tagged files.

Boomerang is designed to scale and perform well under extreme loads. It is designed to operate on multiple processors simultaneously, i.e., with multiple instances of objects and methods. Multiple Web servers and Java virtual machines may be used as server loading and traffic demands.

In one embodiment of the present invention, most of the Boomerang application, Web server 1360 (in one embodiment of the present invention, Stronghold Apache), JRun Servlet Engine 1370, and a Remote Memory Invocation (RMI) registry will run on each of the two Web server machines 110A and 110B (Figure 1). The JRun Servlet Engine 1370, and RMI registry are common Java objects; the RMI API allows a servlet to invoke the methods of a Java object executing on another machine. The Session Manager 1310, another RMI registry, and the Informix database software 1380 will run on one of the databases servers 110C or 110D, the other database server acting as a warm fail-over device.

- *Software Components*

In the following discussion, the terms "Member" and "user" are used interchangeably to refer to a person who has established a login identify in the EBC system. Strictly speaking, however, a Member is a person who has completed the New Member process and properly replied to the authorization email; user is a person who has not.

- *Session Manager 1310*

When the system is first accessed, a session will be created. This session will be identified by a unique session ID. This ID will be used for the following tasks.

Once a user has successfully logged in, the session ID will make a record of this fact so that when a user returns to a page that requires user authentication or identification, the user will not be re-prompted to login. The session ID will be passed from Web page to page as either a hidden field in a form or by rewriting the URL, both well-known techniques in the art. Alternatively, the user may be given the option of saving authentication information in a local file, known in the art as a cookie. The system will check for this cookie prior to displaying the login page.

The session ID will expire if it has not been accessed in a time configured by the EBC system administrator.

- *Login Servlet 1320*

This servlet is activated when a function requiring user authentication is called and the user has not yet logged in. The login screen is shown and user name and password input received. The input name and password are matched against a users table. If the user name is found and the proper password supplied, the request (and session ID, if present) is passed to the Search Servlet.

- *Search Servlet 1330*

The Search Servlet requests a session ID if one was not passed to it, or validates the ID if one was passed. If the requested operation requires user authentication and the user has not already logged in, the Login Servlet is called. All URL and form parameters are passed to the application logic of the servlet for processing.

After processing the URL and form parameters, a HTML document (e.g., a results screen) is returned to the requestor, typically the browser window.

- *JDBC Objects/Classes 1340*

These consist of generic JDBC classes that execute queries and return results in a Java hash table indexed by column name. To make more efficient use of database resources, all structured query language (SQL) statements are prepared at servlet initialization.

Also at servlet initialization, database connection pool 1390 creates a number of connections to the database, the precise number of which is configurable by the EBC system administrator. These connections are utilized by each servlet thread on an as-needed basis. Additional connections are created as demand requires; inactive connections are periodically recycled.

- *HTML Template Engine* \

All pages displayed by the Boomerang application, including the help and information screens, are dynamically generated. The base HTML code and image links for these pages are stored as template files which are preloaded on servlet initialization. These files are parsed and custom tags replaced with data extracted from the database (or calculated) before sending the page to the requestor and display to the user.

- *Administration Programs*

In addition to the Boomerang application, the EBC system also includes a set of shell and SQL scripts that operate on a periodic basis (i.e., are activated by `cron`) to perform housekeeping and maintenance tasks. In particular, the following program functions are required.

a) Purge inactive users who have not activated themselves via the login email after a defined period of time, for example one month

b) Expire "Where Am I?" temporary contact information

c) Lock and unlock IP addresses (to prevent access by IP addresses that appear to be attempting to access the service for illicit purposes)

d) Rollover logs and reports

One of ordinary skill in the art will recognize that these functions may be implemented in a single program or script or in a set of programs and/or scripts. Accordingly, the present invention is not limited by the method of implementation of these functions.

- *Database Schema*

Figures 19A, 19B, and 19C illustrate the database structure and one-to-many /one-to- one relationships between records and fields in one embodiment of the present invention. In particular, relationship symbol 1410 denotes a one-to-many correspondence, the "one" side represented by the "=" sign end. See Figure 19A for an

example. Relationship symbol 1420 (see Figure 19B) denotes a one-to-one mapping, the "one" side again represented by the "=" sign end.

The notations PK, FK1, and I1 – I5 represent SQL field labels. SERIAL, in particular, is a well-known data type.

Although a particular database schema and naming convention for records and fields is described, those skilled in the art will realize that schema other than that described can also be used. Accordingly, the invention is not limited to any particular type of database schema.

Alternate Embodiments

In an alternate embodiment, the EBC system may be installed behind a conventional network "firewall" security device and thus made accessible only to browsers connected to and authorized to use the intranet defined by and behind the firewall.

Access to and delivery of contact information in the EBC system is also not limited to a Web browser interface as commonly known today. Some embodiments of the present invention provide multi-mode access interfaces, including but not limited to interfaces using voice-controlled and conventional wireless PDA and/or cell phones, two-way pagers, and wireless air protocol (WAP)-enabled devices.

Further embodiments of the present invention provide data delivery interface embodiments using, for example, the common alphanumeric pager, wireless markup language (WML), or voice delivery (e.g., audio playback) systems commonly used in the art.

The EBC system also provides an advanced search function that allows users to search for records matching specific, desired characteristics. The search can be made based on one or more of a variety of database parameters, including but not limited to field value (e.g., NAME = Sheehy), date of entry, or a Boolean combination of search terms.

In an alternate embodiment of the "Where Am I?" location tracking feature, the cardholder can rapidly designate one of a pre-defined set of contact locations described by meatspace address, phone number, FAX number, and/or email address.

The order in which the steps of the processes depicted above are performed is purely illustrative in nature. In fact, the steps in each process flow can be performed in any order or in parallel, unless otherwise indicated by the present disclosure.

The method of the present invention may be performed in either hardware, software, or any combination thereof, as those terms are currently known in the art. In particular, the present method may be carried out by software, firmware, or microcode operating on a computer or computers of any type. Additionally, software embodying the present invention may comprise computer instructions in any form (e.g., source code, object code, or interpreted code) stored in any computer-readable medium (e.g., ROM, RAM, magnetic media, punched tape or card, compact disc (CD) in any form, DVD). Furthermore, such software may also be in the form of a computer data signal embodied in a carrier wave, such as that found within the well-known Web pages transferred among computers connected to the Internet. Accordingly, the present invention is not limited to any particular platform, unless specifically stated otherwise in the present disclosure.

While particular embodiments of the present invention have been shown and described, it will be apparent to those skilled in the art that changes and modifications may be made without departing from this invention in its broader aspect and, therefore, the appended claims are to encompass within their scope all such changes and modifications as fall within the true spirit of this invention.

Trademark Notices

Cisco Systems and PIX are registered trademarks and Catalyst, Local Director are trademarks of Cisco Systems, Inc., San Jose California.

Hewlett-Packard and HP-UX are registered trademarks and MC-Service Guard and ecardfile are trademarks of the Hewlett-Packard Company, Palo Alto, California.

Informix is a registered trademark and Informix Dynamic Server is a trademark of Informix Software, Menlo Park, California.

Java is a trademark of Sun Microsystems, Inc., Palo Alto, California.

JRun is a trademark of Allure Corp., Cambridge, Massachusetts.

Microsoft is a registered trademark of Microsoft Corp., Redmond, Washington.

Netscape, Netscape Navigator, and Netscape Communicator are registered trademarks of Netscape Communications Corp., Mountain View, California.

Stronghold is a registered trademark of C2Net, Inc., Oakland, California.

CLAIMS

We claim:

1. A method of providing access to a collection of electronic business cards comprising the steps of:

providing an electronic business card file Web site to a user;

allowing the creation of an electronic business card file by the user using said Web site, said creation comprising:

allowing the user to enter information into a plurality of fields;

storing said information; and

sending an authentication email to the user, wherein a reply to said authentication email is required to complete said creation;

allowing the user to search for one or more records;

allowing the user to view said records; and

if said creation is completed, allowing the setting of privacy levels by the user for each said field, said setting comprising selecting one of more than two privacy levels.

2. The method of Claim 1, wherein said electronic business card file Web site is accessible from the Internet.

3. The method of Claim 1, wherein said electronic business card file Web site is accessible from an intranet isolated from the Internet by a firewall security device.

4. The method of Claim 1, wherein said electronic business card file Web site is accessible from a web browser.

5. The method of Claim 1, wherein said electronic business card file Web site is accessible from a personal digital assistant.

6. The method of Claim 1, wherein said electronic business card file Web site is accessible from a browser-enabled telephone.

7. The method of Claim 1, wherein said electronic business card file Web site is accessible by spoken commands.

8. The method of Claim 1, wherein said first format comprises audio playback.

9. A computer system for providing access to a collection of electronic business cards, comprising computer instructions for:

providing an electronic business card file Web site to a user;

allowing the creation of an electronic business card file by the user using said Web site, said creation comprising:

allowing the user to enter information into a plurality of fields;

storing said information; and

sending an authentication email to the user, wherein a reply to said authentication email is required to complete said creation;

allowing the user to search for one or more records;

allowing the user to view said records; and

if said creation is completed, allowing the setting of privacy levels by the user for each said field, said setting comprising selecting one of more than two privacy levels.

10. The computer system of Claim 9, wherein said electronic business card file Web site is accessible from the Internet.

11. The computer system of Claim 9, wherein said electronic business card file Web site is accessible from an intranet isolated from the Internet by a firewall security device.

12. The computer system of Claim 9, wherein said electronic business card file Web site is accessible from a web browser.

13. The computer system of Claim 9, wherein said electronic business card file Web site is accessible from a personal digital assistant.

14. The computer system of Claim 9, wherein said electronic business card file Web site is accessible from a browser-enabled telephone.

15. The computer system of Claim 9, wherein said electronic business card file Web site is accessible by spoken commands.

16. The computer system of Claim 9, wherein said first format comprises audio playback.

17. A computer-readable storage medium, comprising computer instructions for:

providing an electronic business card file Web site to a user;

allowing the creation of an electronic business card file by the user using said Web site, said creation comprising:

allowing the user to enter information into a plurality of fields;

storing said information; and

sending an authentication email to the user, wherein a reply to said authentication email is required to complete said creation;

allowing the user to search for one or more records;

allowing the user to view said records; and

if said creation is completed, allowing the setting of privacy levels by the user for each said field, said setting comprising selecting one of more than two privacy levels.

18. The computer-readable storage medium of Claim 17, wherein said electronic business card file Web site is accessible from the Internet.

19. The computer-readable storage medium of Claim 17, wherein said electronic business card file Web site is accessible from an intranet isolated from the Internet by a firewall security device.

20. The computer-readable storage medium of Claim 17, wherein said electronic business card file Web site is accessible from a web browser.

21. The computer-readable storage medium of Claim 17, wherein said electronic business card file Web site is accessible from a personal digital assistant.

22. The computer-readable storage medium of Claim 17, wherein said electronic business card file Web site is accessible from a browser-enabled telephone.

23. The computer-readable storage medium of Claim 17, wherein said electronic business card file Web site is accessible by spoken commands.

24. The computer-readable storage medium of Claim 17, wherein said first format comprises audio playback.

25. A computer data signal embodied in a carrier wave, comprising computer instructions for:

providing an electronic business card file Web site to a user;

allowing the creation of an electronic business card file by the user using said Web site, said creation comprising:

allowing the user to enter information into a plurality of fields;

storing said information; and

sending an authentication email to the user, wherein a reply to said authentication email is required to complete said creation;

allowing the user to search for one or more records;

allowing the user to view said records; and

if said creation is completed, allowing the setting of privacy levels by the user for each said field, said setting comprising selecting one of more than two privacy levels.

26. The computer data signal of Claim 25, wherein said electronic business card file Web site is accessible from the Internet.

27. The computer data signal of Claim 25, wherein said electronic business card file Web site is accessible from an intranet isolated from the Internet by a firewall security device.

28. The computer data signal of Claim 25, wherein said electronic business card file Web site is accessible from a web browser.

29. The computer data signal of Claim 25, wherein said electronic business card file Web site is accessible from a personal digital assistant.

30. The computer data signal of Claim 25, wherein said electronic business card file Web site is accessible from a browser-enabled telephone.

31. The computer data signal of Claim 25, wherein said electronic business card file Web site is accessible by spoken commands.

32. The computer data signal of Claim 25, wherein said first format comprises audio playback.

E-SERVICE TO MANAGE CONTACT INFORMATION WITH PRIVACY LEVELS

James G. Douvikas
Terry R. Sheehy
Chris W. T. McKay

ABSTRACT OF THE DISCLOSURE

A method of providing an electronic business card (EBC) access and organization service on the Web. The cardholder database is accessible and searchable from any browser connected to the Internet or the EBC service may be installed behind a conventional firewall and thus accessible only to intranet users. The service thus provides easy access to cardholder contact information with privacy assured by use of integrated access restrictions. Access to and delivery of contact information by the service is not limited to a Web browser interface as commonly known today. The service provides multi-mode access and/or data delivery interfaces. The service also provides an export feature that formats search results into a pre-defined file structure readable by a conventional contact management programs. Custom export file formats may also be defined provide even wider connectivity and cross-platform utility. Access to individual records is controlled at both the record level and the field level, with multiple privacy levels for each field, in addition to the well-known "public" and "private" levels. Users having certain permissions are permitted to read a defined group of records, though not necessarily all fields in each record. A location tracking feature is also provided to allow the cardholder to rapidly designate a pre-defined contact location. Alternately, the cardholder may define a temporary contact location not normally stored in the database system. Electronic mail sent by the cardholder is automatically formatted to contain a signature hypertext link directing recipients of the email to the EBC service. This hyperlink enables the recipient of the email to rapidly access the EBC system to locate the cardholder and/or obtain additional information.

FIG. 1

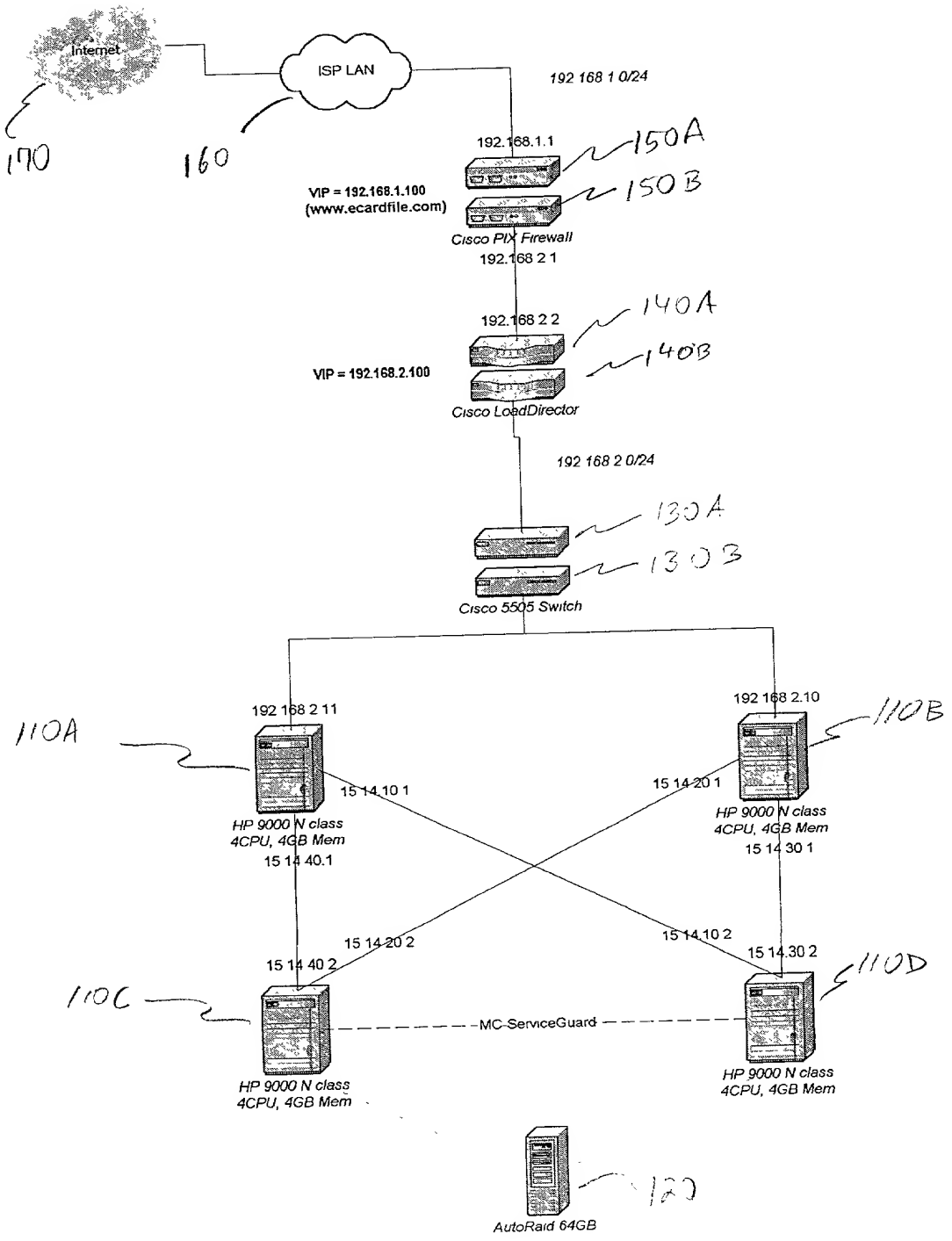

Internet
ISP LAN
192 168 1 0/24
170
160
192.168.1.1
150A
150B
VIP = 192.168.1.100
(www.ecardfile.com)
Cisco PIX Firewall
192.168 2 1
192.168 2 2
140A
140B
VIP = 192.168.2.100
Cisco LoadDirector
192 168 2 0/24
130A
130B
Cisco 5505 Switch
192 168 2 11
110A
192 168 2.10
110B
15 14.10 1
15 14.20 1
HP 9000 N class
4CPU, 4GB Mem
15 14 40.1
HP 9000 N class
4CPU, 4GB Mem
15 14 30 1
15 14.20 2
15 14.10 2
15 14 40 2
15 14.30 2
110C
110D
MC-ServiceGuard
HP 9000 N class
4CPU, 4GB Mem
HP 9000 N class
4CPU, 4GB Mem
120
AutoRaid 64GB

FIG. 2

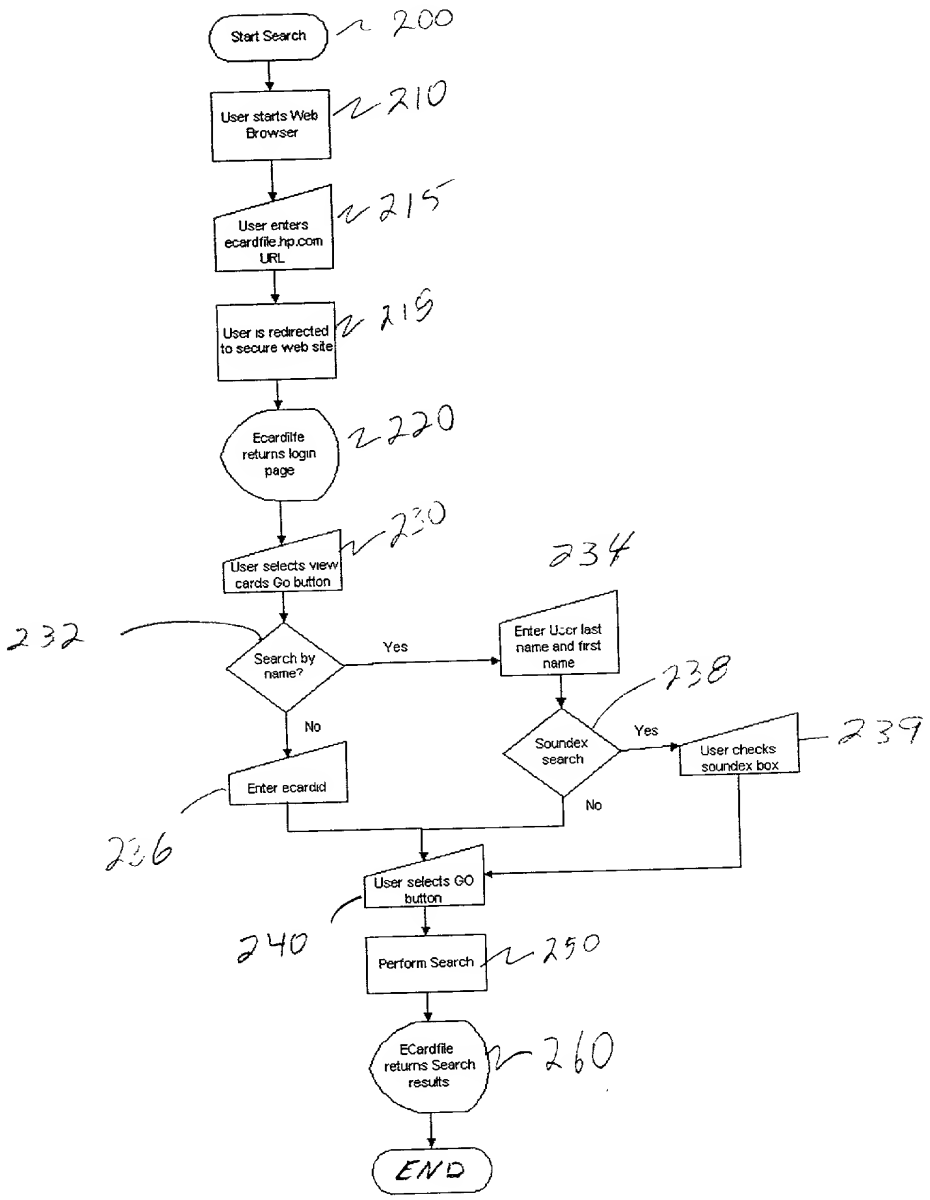

Start Search
200
User starts Web Browser
210
User enters ecardfile.hp.com URL
215
User is redirected to secure web site
219
Ecardlife returns login page
220
User selects view cards Go button
230
234
232
Search by name?
Yes
Enter User last name and first name
238
Soundex search
Yes
User checks soundex box
239
No
Enter ecardid
No
236
User selects GO button
240
Perform Search
250
ECardfile returns Search results
260
END

FIG. 3A

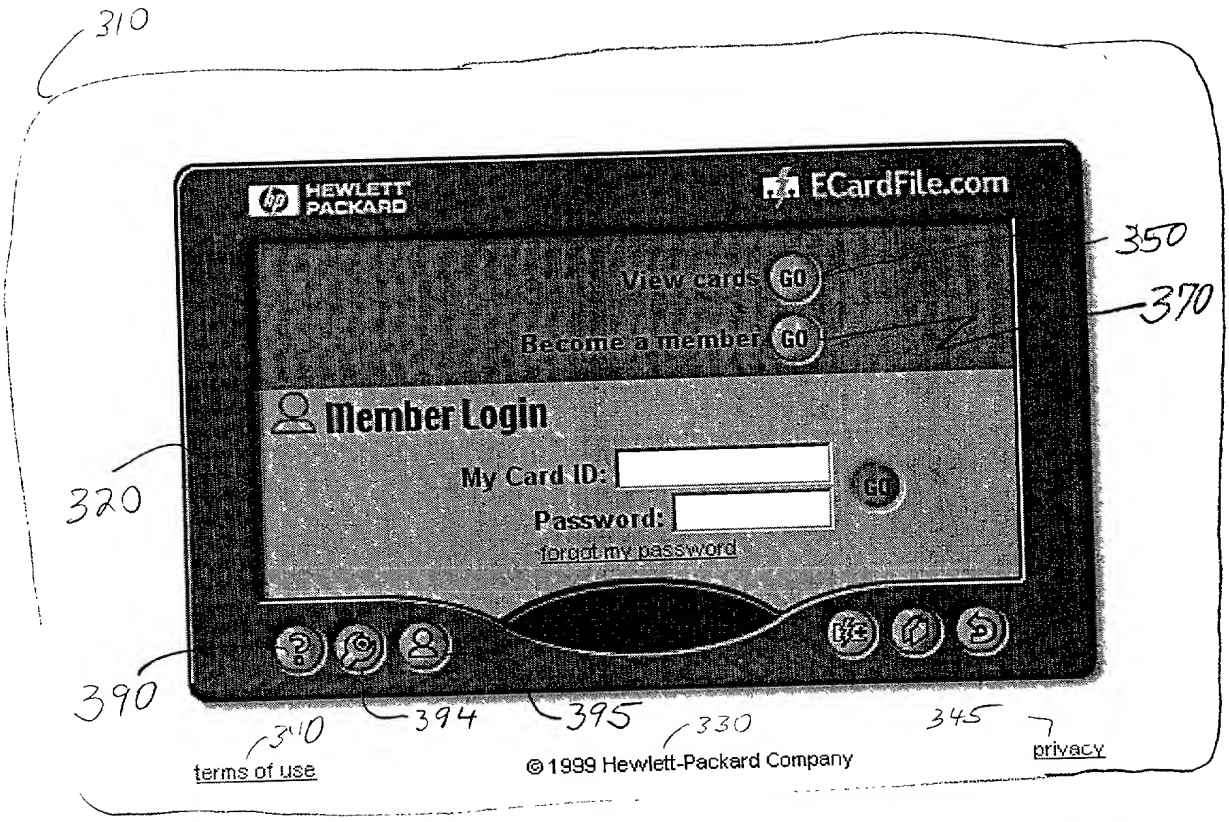

310
HEWLETT PACKARD
ECardFile.com
View cards GO
Become a member GO
350
370
Member Login
My Card ID:
Password:
GO
forgot my password
320
390
394
395
330
345
310
terms of use
©1999 Hewlett-Packard Company
privacy

FIG. 3B

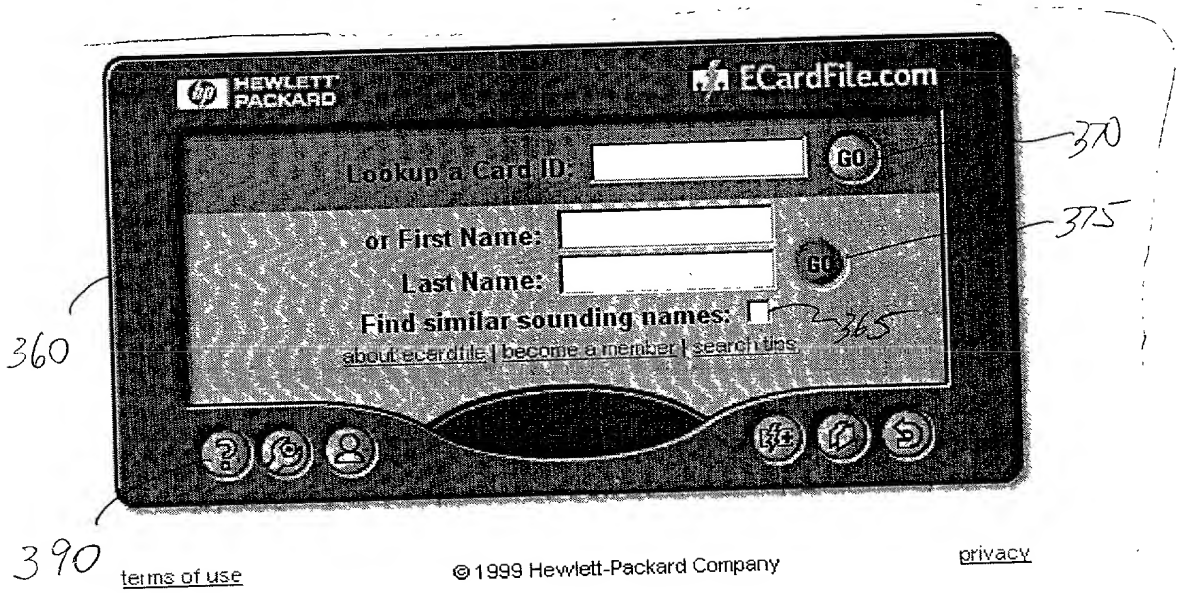
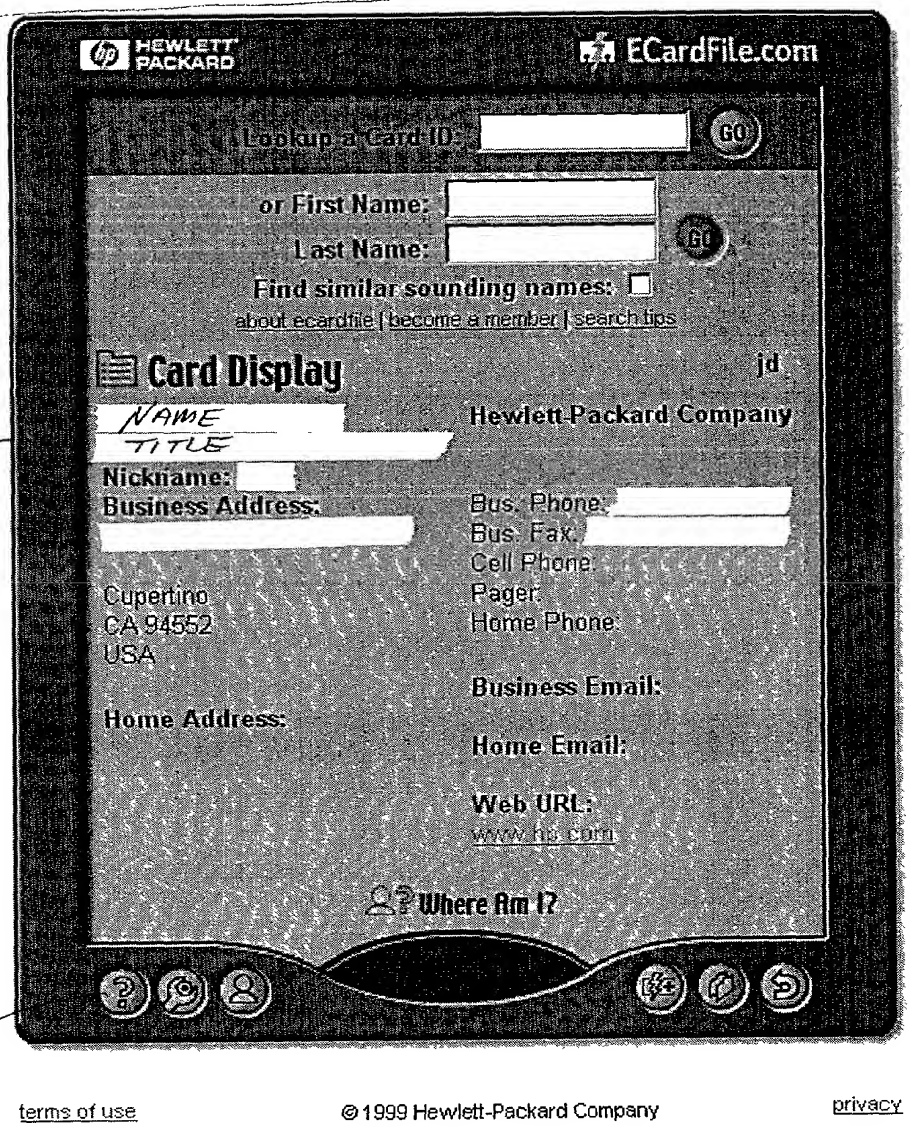

310
HEWLETT PACKARD
ECardFile.com
Lookup a Card ID:
GO
370
or First Name:
Last Name:
GO
375
Find similar sounding names:
365
about ecardfile | become a member | search tips
360
390
terms of use
©1999 Hewlett-Packard Company
privacy

FIG. 4



310

410

390

FIG. 5

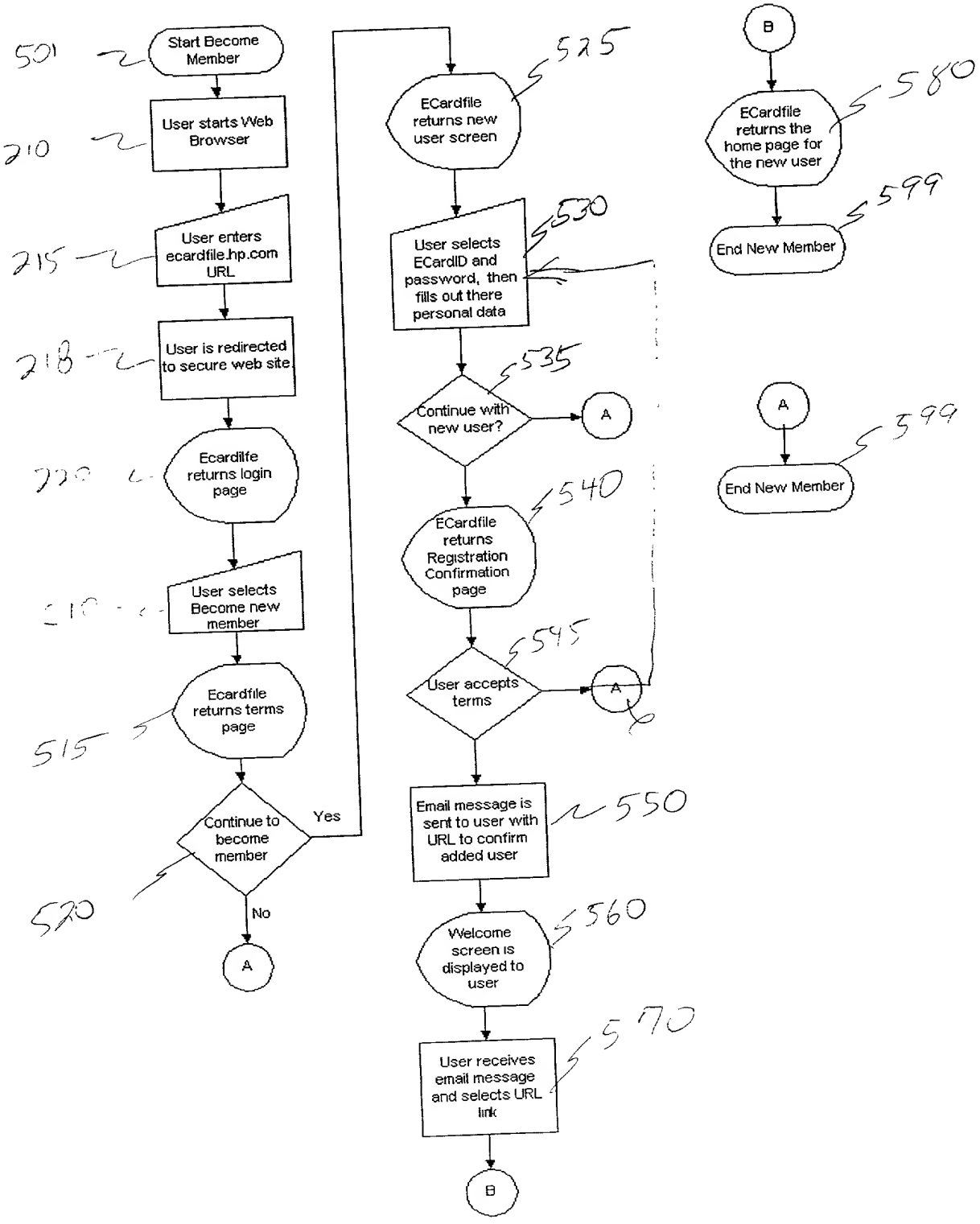

501
Start Become Member
210
User starts Web Browser
215
User enters ecardfile.hp.com URL
218
User is redirected to secure web site
220
Ecardilfe returns login page
510
User selects Become new member
515
Ecardfile returns terms page
520
Continue to become member
Yes
No
A
525
ECardfile returns new user screen
530
User selects ECardID and password, then fills out there personal data
535
Continue with new user?
A
540
ECardfile returns Registration Confirmation page
545
User accepts terms
A
550
Email message is sent to user with URL to confirm added user
560
Welcome screen is displayed to user
570
User receives email message and selects URL link
B
B
580
ECardfile returns the home page for the new user
599
End New Member
A
599
End New Member

FIG. 6

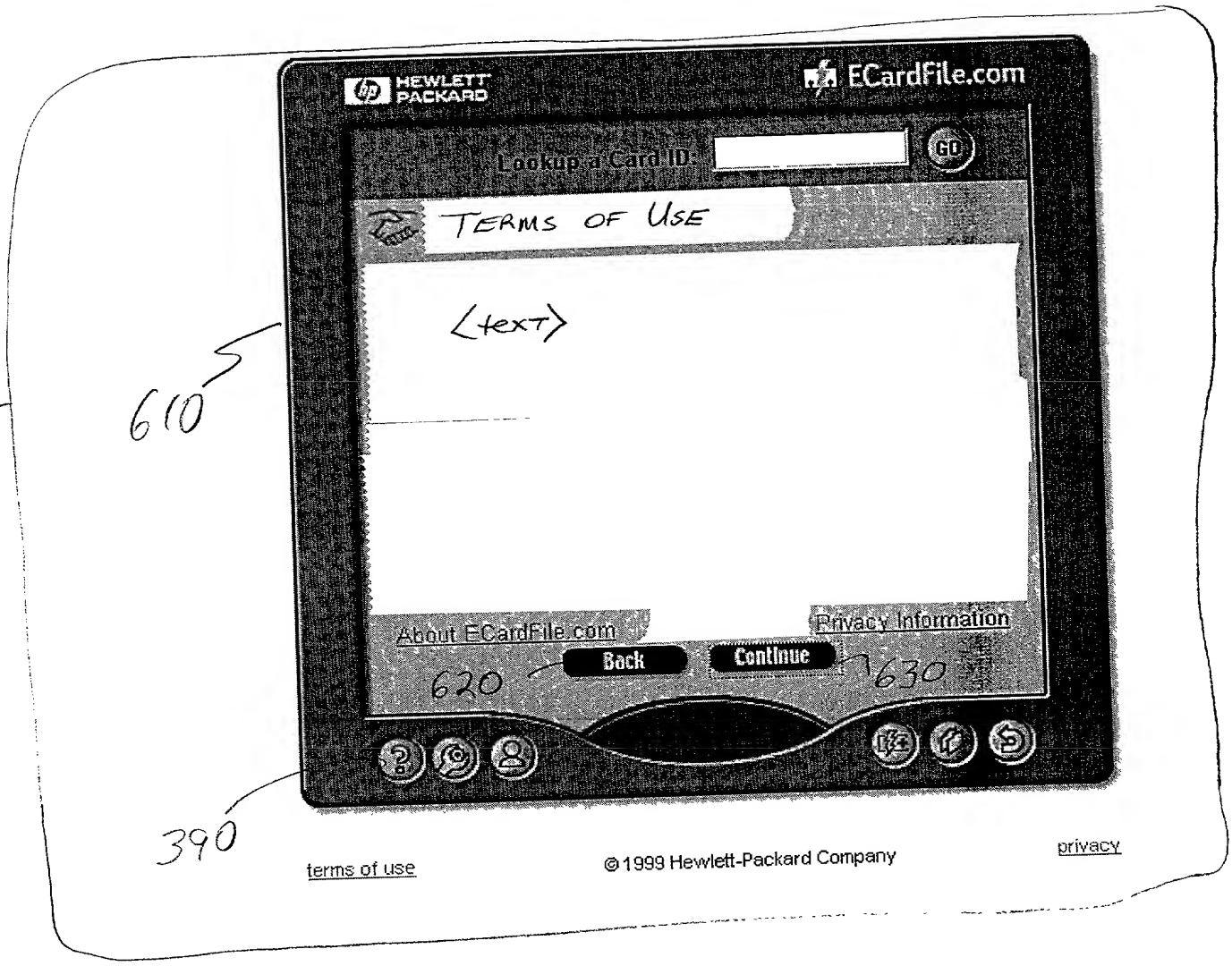
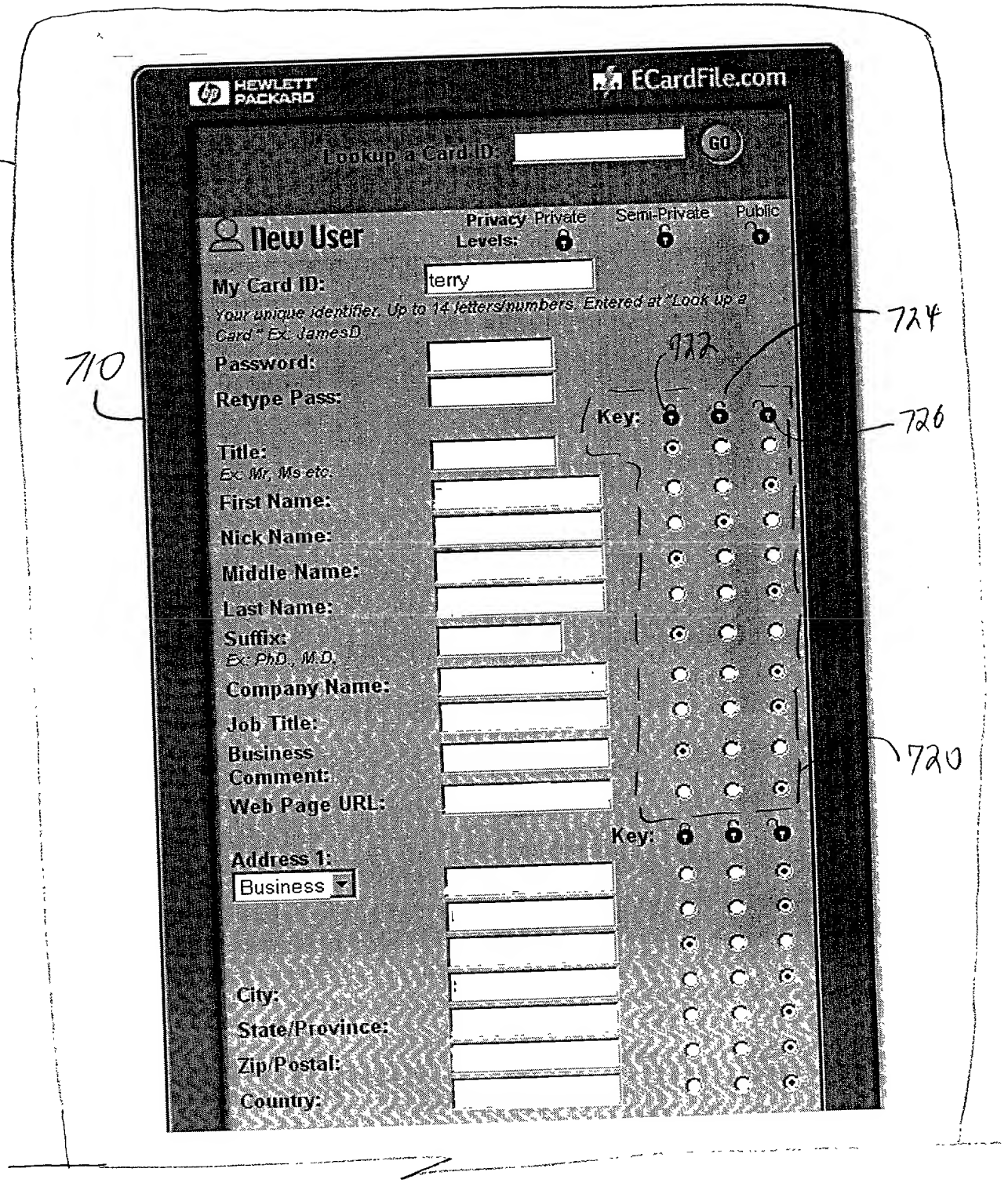

HEWLETT PACKARD
ECardFile.com
Lookup a Card ID:
GO
TERMS OF USE
<text>
610
610
About ECardFile.com
Privacy Information
Back
Continue
620
630
390
terms of use
©1999 Hewlett-Packard Company
privacy

FIG. 7A



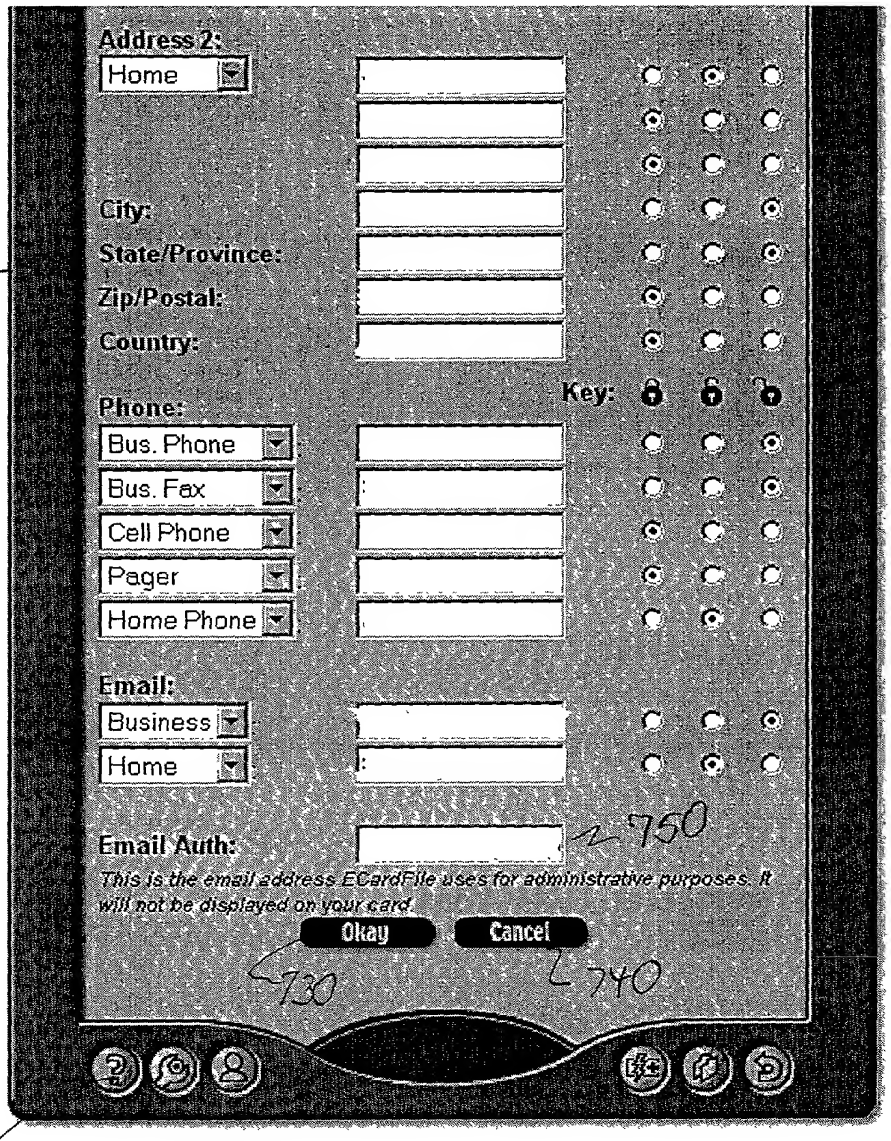
LEGEND

Fig. 7A
Fig. 7B

Fig. 7B

310

New User Bottom:



710

750

730

740

390

terms of use © 1999 Hewlett-Packard Company privacy

LEGEND

Fig. 7A

Fig. 7B

FIG. 8

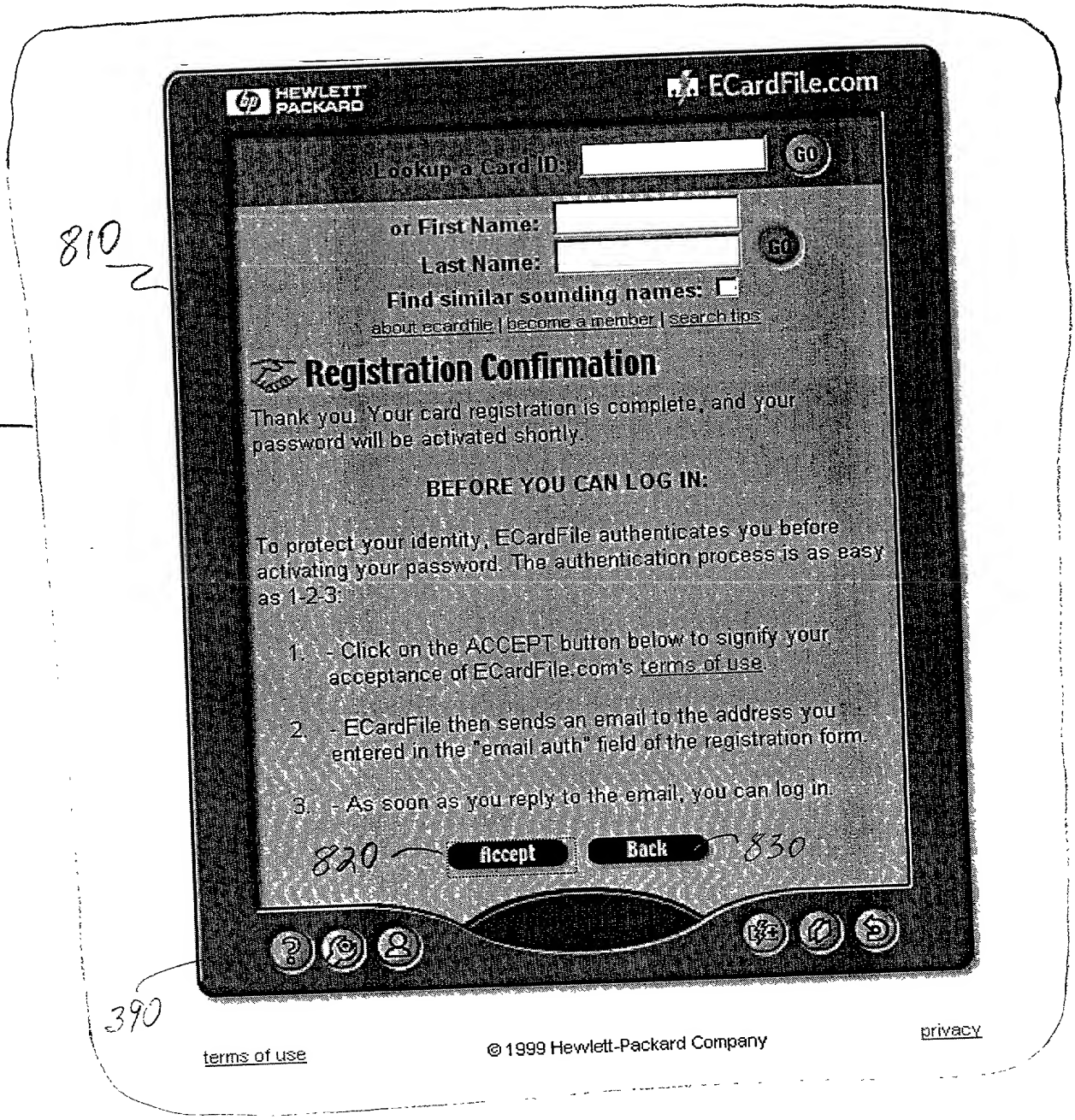
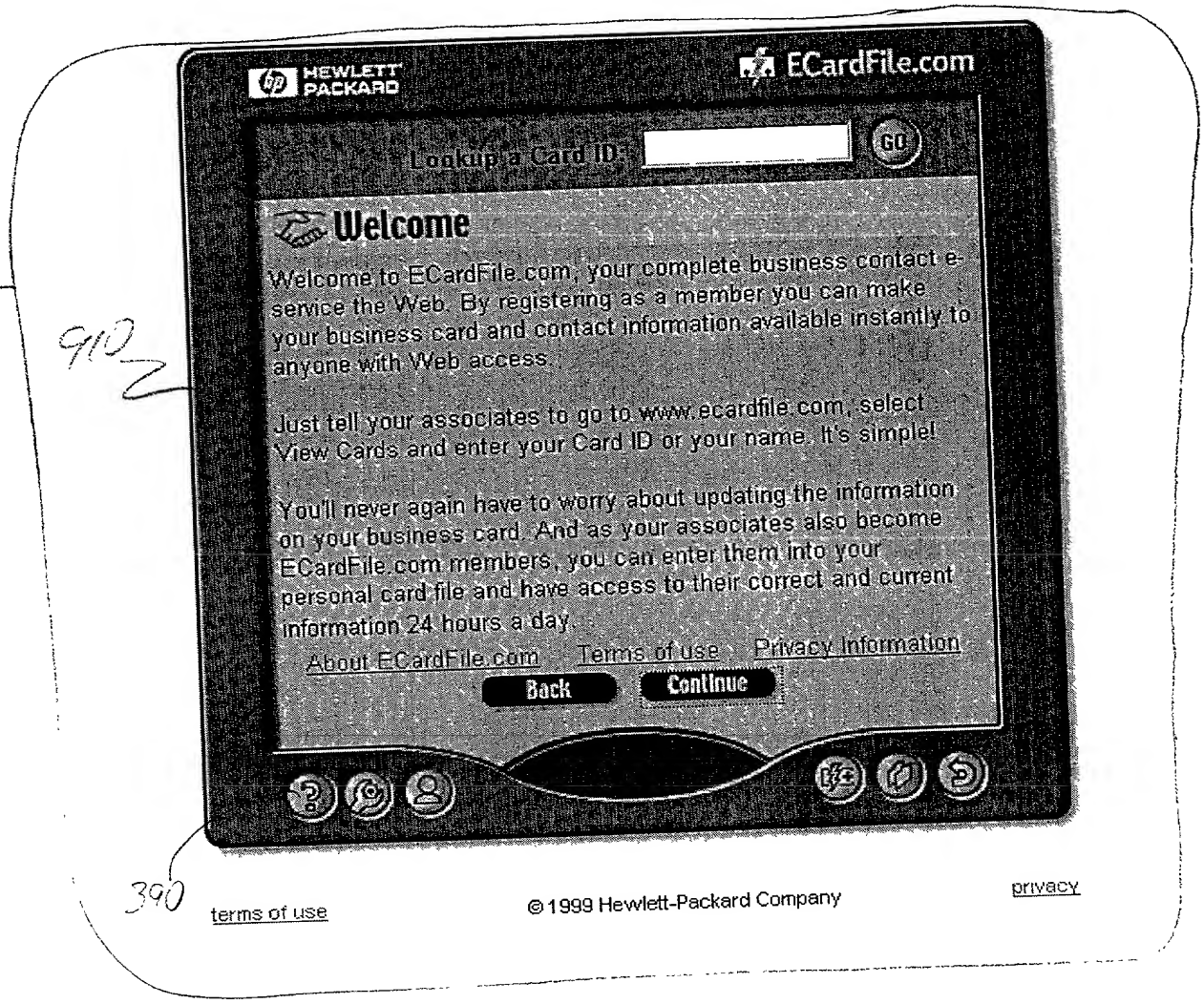

HEWLETT PACKARD
ECardFile.com
Lookup a Card ID:
GO
or First Name:
Last Name:
GO
Find similar sounding names:
about ecardfile | become a member | search tips
Registration Confirmation
Thank you. Your card registration is complete, and your password will be activated shortly.
BEFORE YOU CAN LOG IN:
To protect your identity, ECardFile authenticates you before activating your password. The authentication process is as easy as 1-2-3:
1. - Click on the ACCEPT button below to signify your acceptance of ECardFile.com's terms of use.
2. - ECardFile then sends an email to the address you entered in the "email auth" field of the registration form.
3. - As soon as you reply to the email, you can log in.
Accept
Back
810
820
830
310
390
terms of use
© 1999 Hewlett-Packard Company
privacy

FIG. 9



310

910

390

FIG. 10

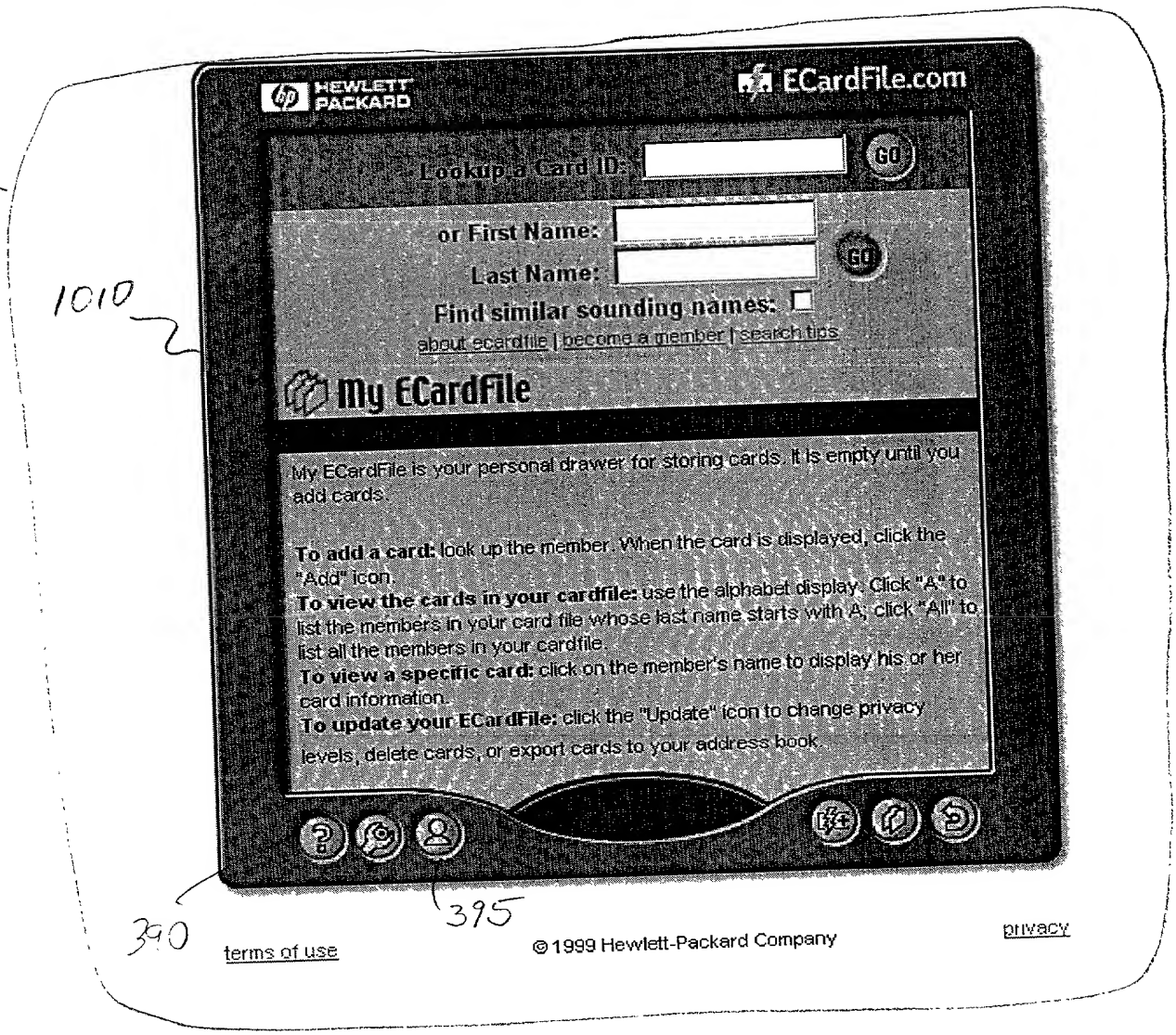


FIG 11

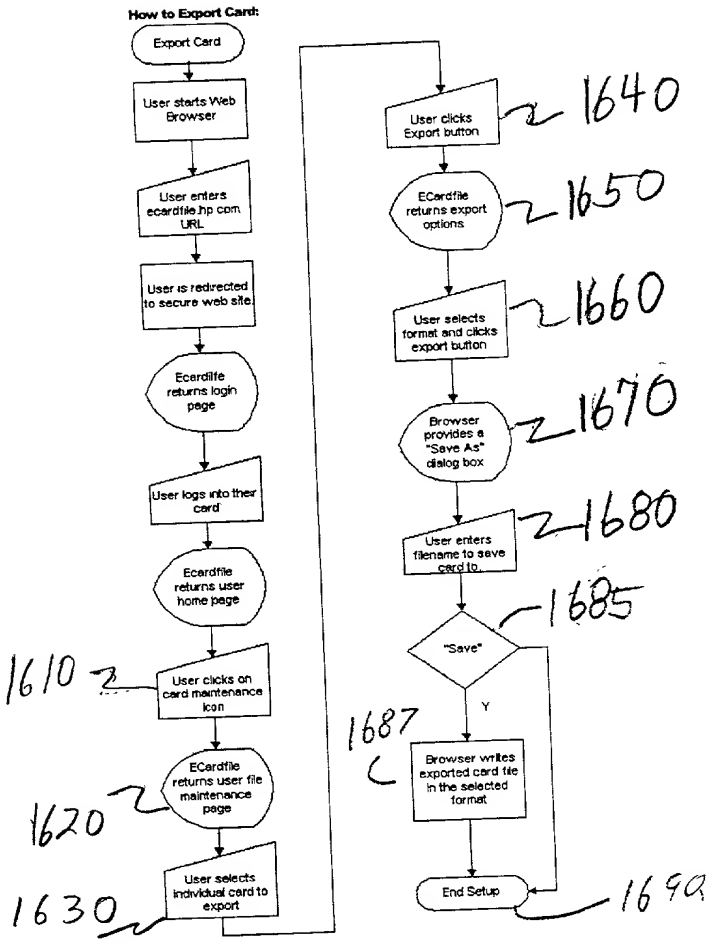
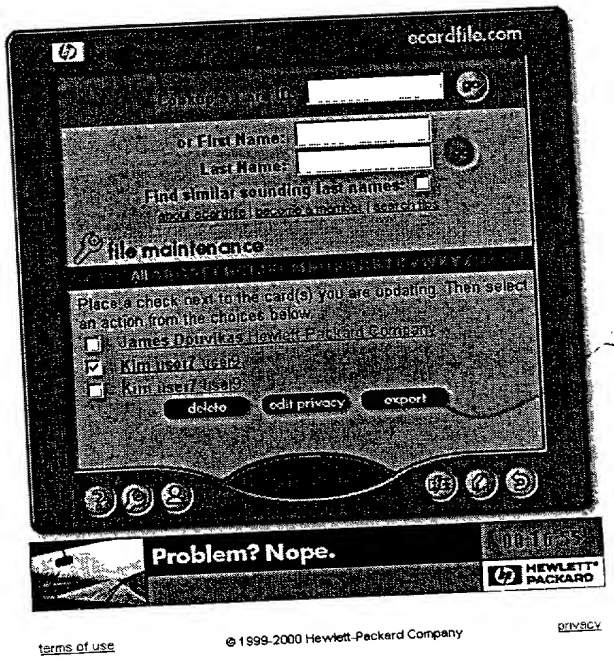


FIG. 12



1710

M-815Y US

FIG. 13

File Export Screen :

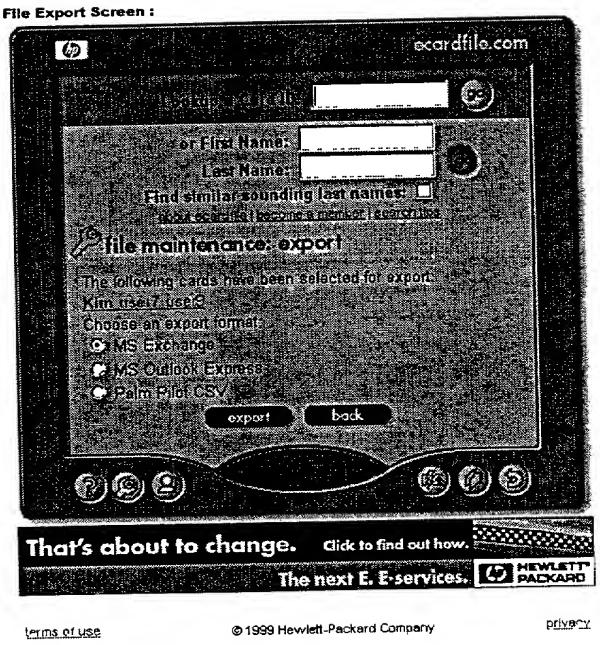


FIG 14

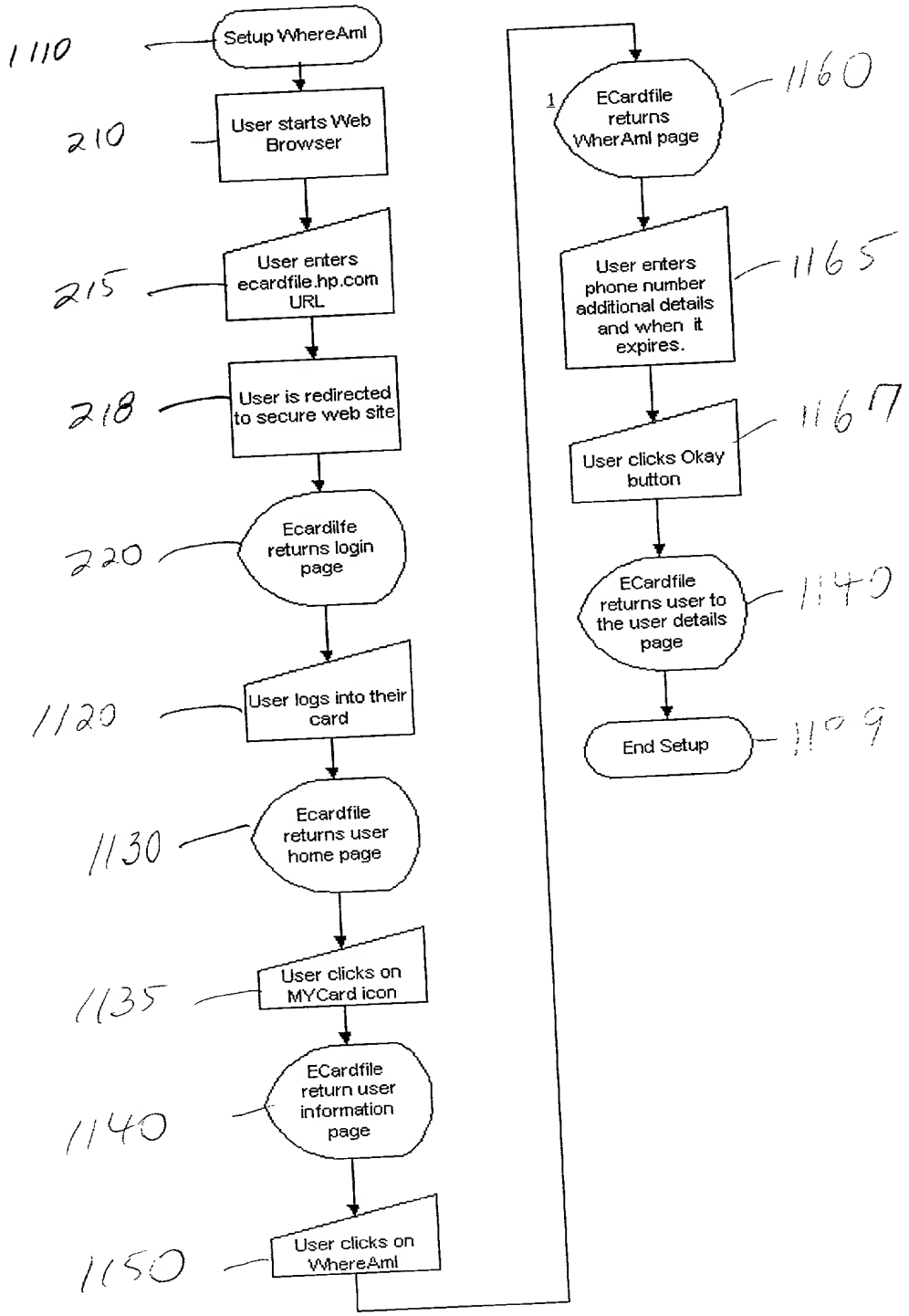
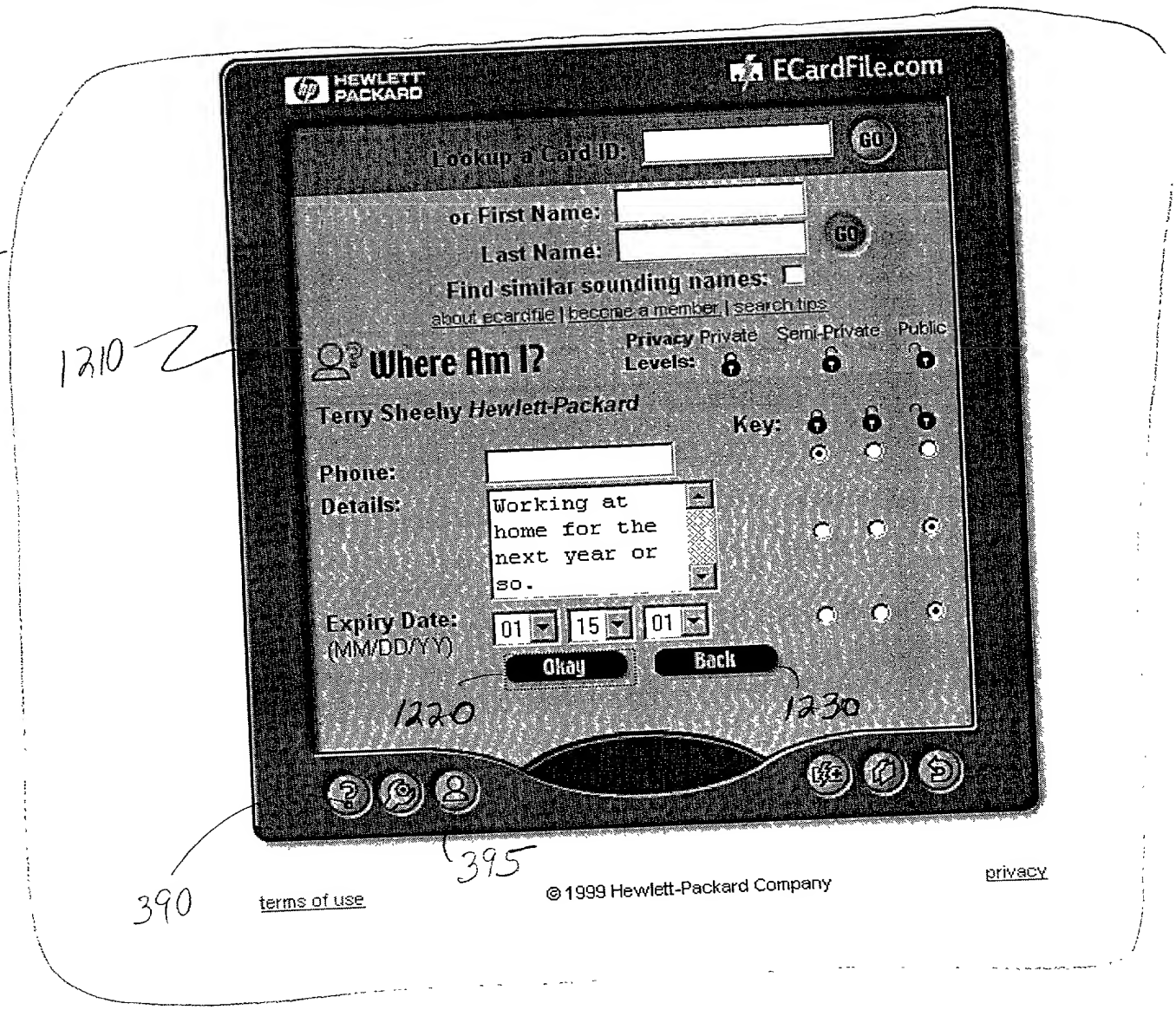

1110
Setup WhereAml
210
User starts Web Browser
215
User enters ecardfile.hp.com URL
218
User is redirected to secure web site
220
Ecardilfe returns login page
1120
User logs into their card
1130
Ecardfile returns user home page
1135
User clicks on MYCard icon
1140
ECardfile return user information page
1150
User clicks on WhereAml
1
1160
ECardfile returns WherAml page
1165
User enters phone number additional details and when it expires.
1167
User clicks Okay button
1170
ECardfile returns user to the user details page
1109
End Setup

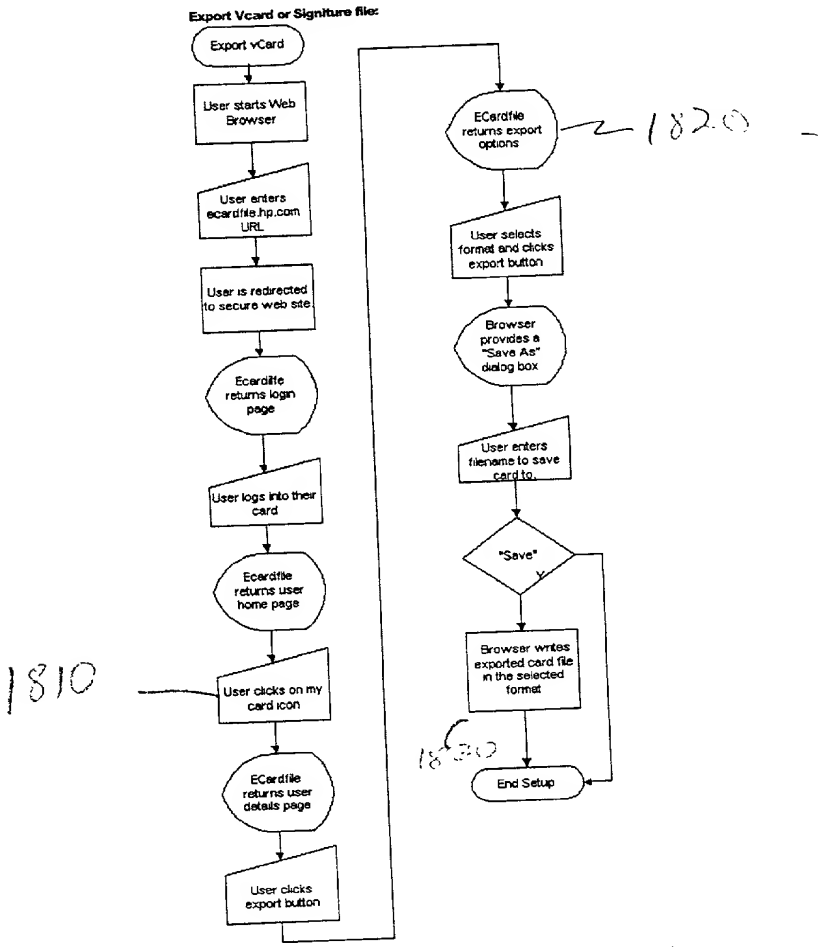
Fig. 15



310-2

1210-2

FIG. 16



M-815X US

FIG 17

Vcard and Signature Export Screen:

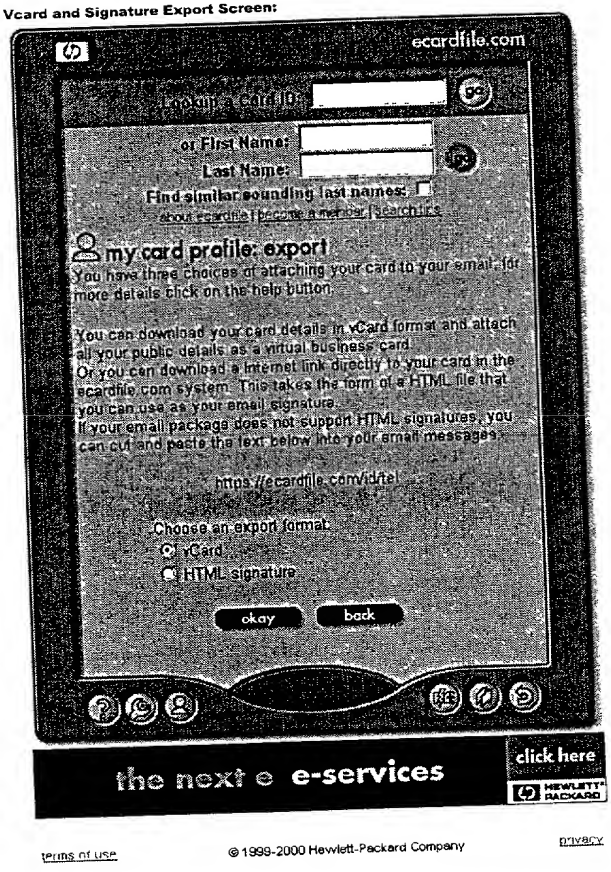


FIG. 18

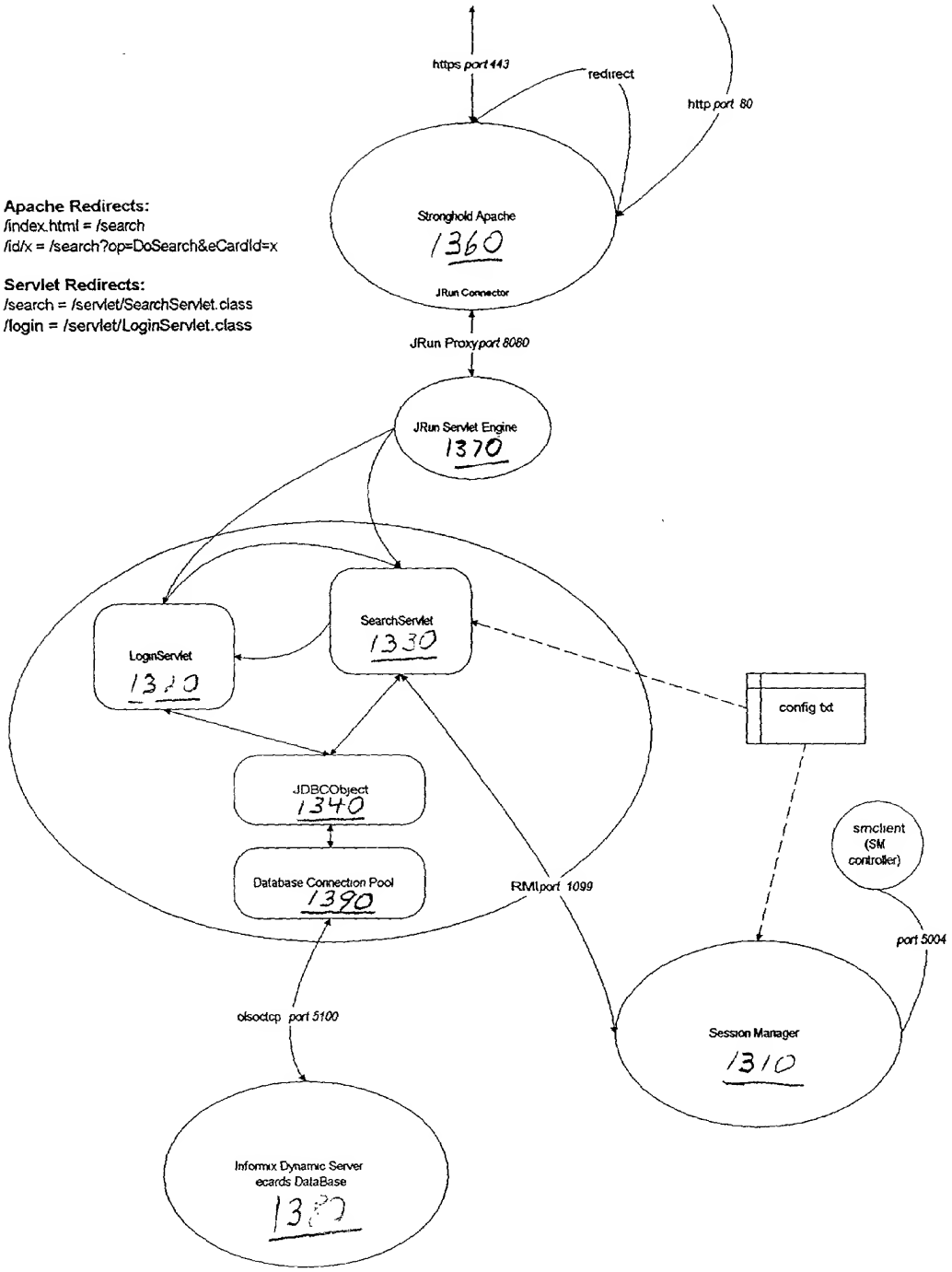

https port 443
redirect
http port 80
Apache Redirects:
/index.html = /search
/id/x = /search?op=DoSearch&eCardId=x
Servlet Redirects:
/search = /servlet/SearchServlet.class
/login = /servlet/LoginServlet.class
Stronghold Apache
1360
JRun Connector
JRun Proxy port 8080
JRun Servlet Engine
1370
LoginServlet
1320
SearchServlet
1330
config.txt
JDBCObject
1340
Database Connection Pool
1390
RMI port 1099
smclient
(SM
controller)
port 5004
olsoctcp port 5100
Session Manager
1310
Informix Dynamic Server
ecards DataBase
1380

FIG. 19A

Database Schema

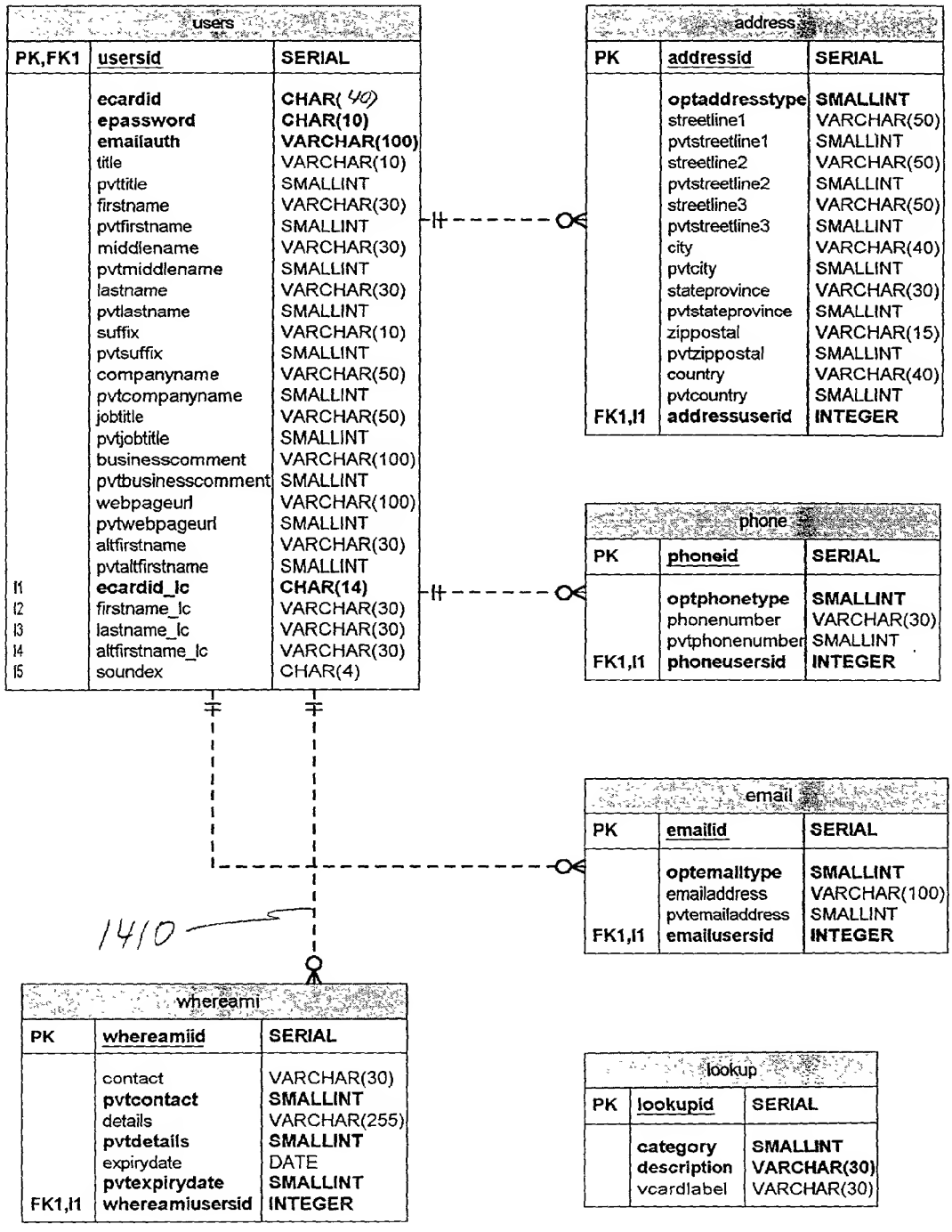
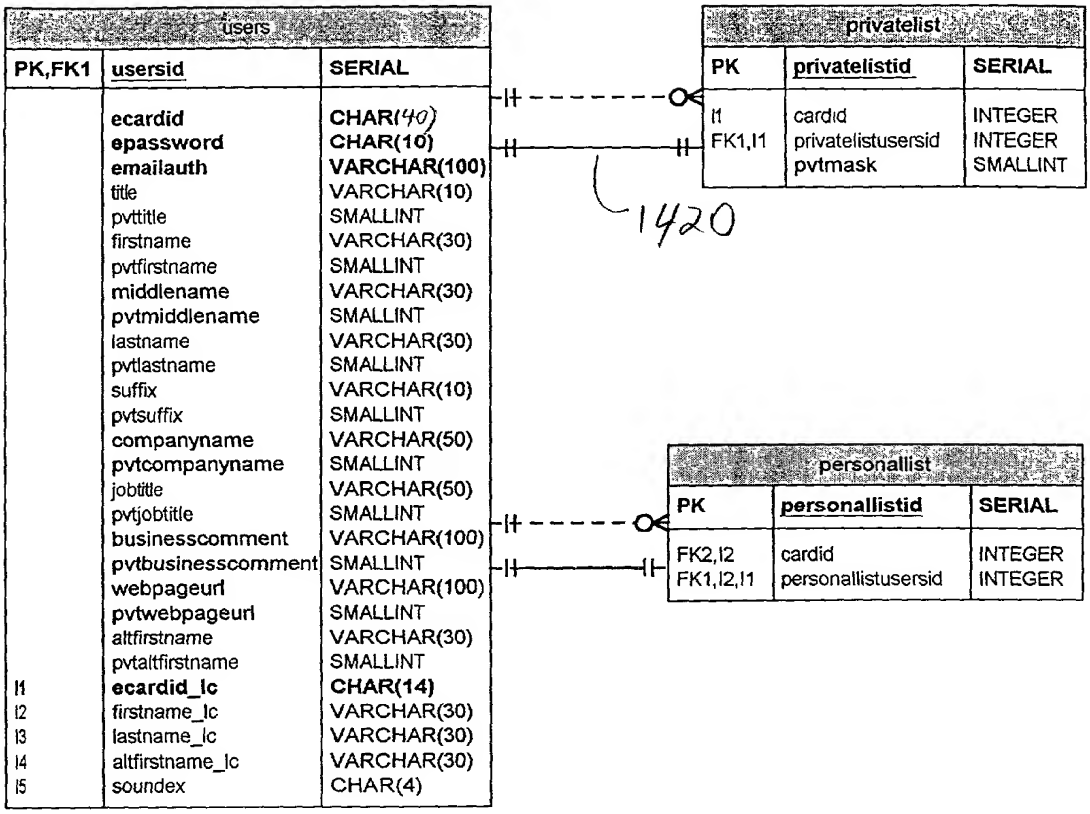
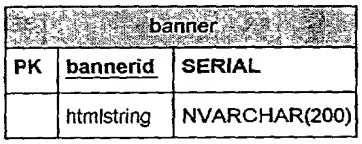


Fig 19B





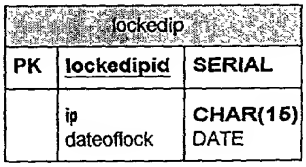
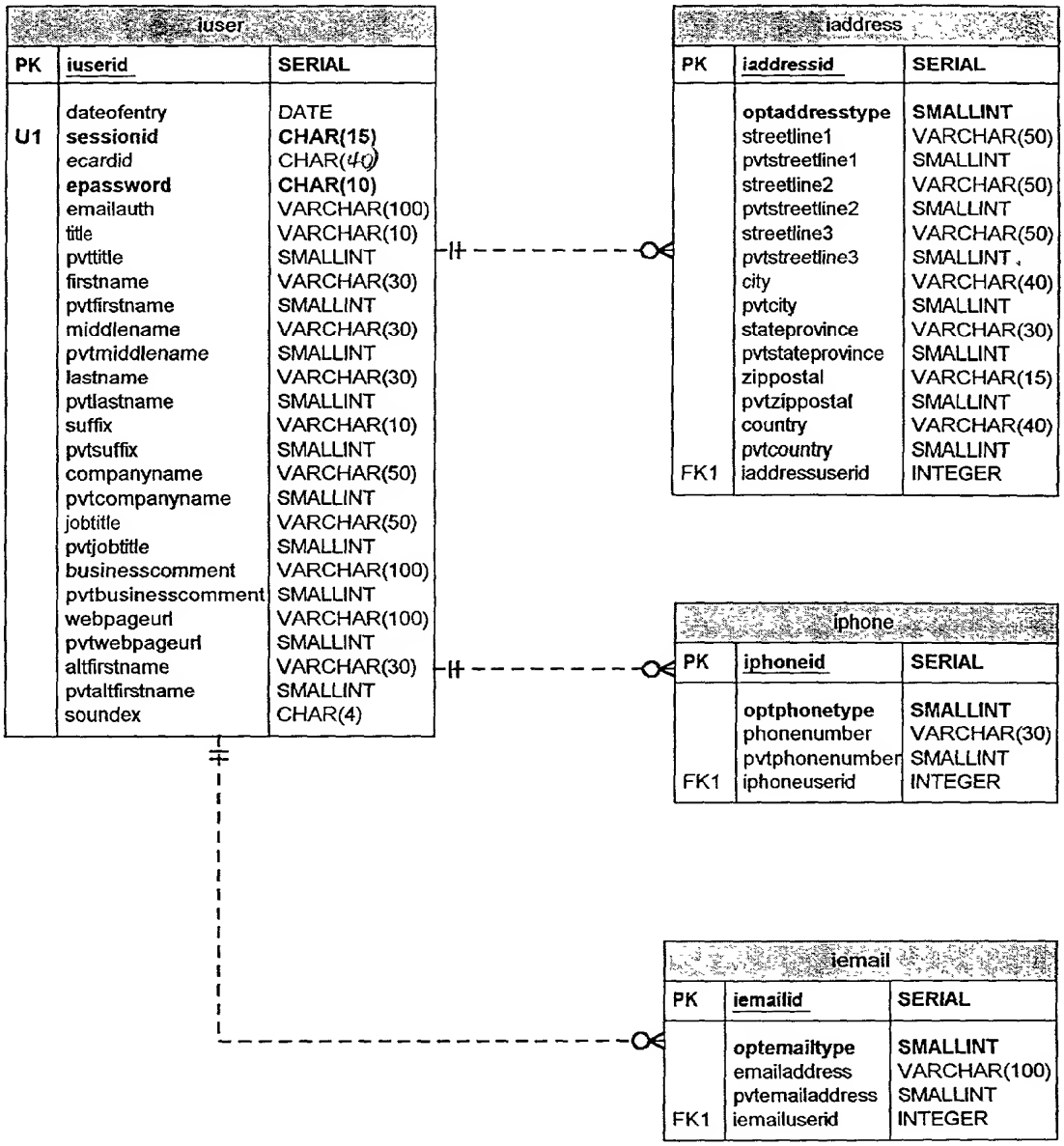


FIG. 19C


iuser
PK iuserid SERIAL
dateofentry DATE
U1 sessionid CHAR(15)
ecardid CHAR(40)
epassword CHAR(10)
emailauth VARCHAR(100)
title VARCHAR(10)
pvttitle SMALLINT
firstname VARCHAR(30)
pvtfirstname SMALLINT
middlename VARCHAR(30)
pvtmiddlename SMALLINT
lastname VARCHAR(30)
pvtlastname SMALLINT
suffix VARCHAR(10)
pvtsuffix SMALLINT
companyname VARCHAR(50)
pvtcompanyname SMALLINT
jobtitle VARCHAR(50)
pvtjobtitle SMALLINT
businesscomment VARCHAR(100)
pvtbusinesscomment SMALLINT
webpageurl VARCHAR(100)
pvtwebpageurl SMALLINT
altfirstname VARCHAR(30)
pvtaltfirstname SMALLINT
soundex CHAR(4)
iaddress
PK iaddressid SERIAL
optaddresstype SMALLINT
streetline1 VARCHAR(50)
pvtstreetline1 SMALLINT
streetline2 VARCHAR(50)
pvtstreetline2 SMALLINT
streetline3 VARCHAR(50)
pvtstreetline3 SMALLINT
city VARCHAR(40)
pvtcity SMALLINT
stateprovince VARCHAR(30)
pvtstateprovince SMALLINT
zippostal VARCHAR(15)
pvtzippostal SMALLINT
country VARCHAR(40)
pvtcountry SMALLINT
FK1 iaddressuserid INTEGER
iphone
PK iphoneid SERIAL
optphonetype SMALLINT
phonenumber VARCHAR(30)
pvtphonenumber SMALLINT
FK1 iphoneuserid INTEGER
iemail
PK iemailid SERIAL
optemailtype SMALLINT
emailaddress VARCHAR(100)
pvtemailaddress SMALLINT
FK1 iemailuserid INTEGER

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

ATTORNEY DOCKET NO. 10992822-1

As a below named inventor, I hereby declare that:

My residence/post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

E-SERVICE TO MANAGE CONTACT INFORMATION WITH PRIVACY LEVELS

the specification of which is attached hereto unless the following box is checked:

( ) was filed on _____ as US Application Serial No. or PCT International Application Number _____ and was amended on _____ (if applicable).

I hereby state that I have reviewed and understood the contents of the above-identified specification, including the claims, as amended by any amendment(s) referred to above. I acknowledge the duty to disclose all information which is material to patentability as defined in 37 CFR 1.56.

Foreign Application(s) and/or Claim of Foreign Priority

I hereby claim foreign priority benefits under Title 35, United States Code Section 119 of any foreign application(s) for patent or inventor(s) certificate listed below and have also identified below any foreign application for patent or inventor(s) certificate having a filing date before that of the application on which priority is claimed:

COUNTRY	APPLICATION NUMBER	DATE FILED	PRIORITY CLAIMED UNDER 35 U.S.C. 119
			YES ___ NO. ___
			YES. ___ NO ___

Provisional Application

I hereby claim the benefit under Title 35, United States Code Section 119(e) of any United States provisional application(s) listed below:

APPLICATION SERIAL NUMBER	FILING DATE

U. S. Priority Claim

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

APPLICATION SERIAL NUMBER	FILING DATE	STATUS (patented/pending/abandoned)

POWER OF ATTORNEY:

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith:

Customer Number 022879

Place Customer Number Bar Code Label here

Send Correspondence to:	Direct Telephone Calls To:
HEWLETT-PACKARD COMPANY Intellectual Property Administration P.O. Box 272400 Fort Collins, Colorado 80528-9599	Paul E. Lewkowicz (408) 453-9200

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Inventor: James G. Douvikas Citizenship: USA

Residence: Danville, California

Post Office Address: 796 Tunbridge Road, Danville, CA 94526

James G. Douvikas 2/18/00

Inventor's Signature Date

DECLARATION AND POWER OF ATTORNEY
FOR PATENT APPLICATION (continued)

ATTORNEY DOCKET NO. 10992822-1

Full Name of # 2 joint inventor: Terry R. Sheehy Citizenship: Britain

Residence: Mountain View, California

Post Office Address: 435 Nicholas Drive, Mountain View, CA 94043

Inventor's Signature Date FEB 18th 00

Full Name of # 3 joint inventor: Christopher W. T. McKay Citizenship: New Zealand

Residence: Paddington, NSW 2021, Australia

Post Office Address: 43 Gordon Street, Paddington, NSW 2021, Australia

Inventor's Signature Date

Full Name of # 4 joint inventor: Citizenship:

Residence:

Post Office Address:

Inventor's Signature Date

Full Name of # 5 joint inventor: Citizenship:

Residence:

Post Office Address:

Inventor's Signature Date

Full Name of # 6 joint inventor: Citizenship:

Residence:

Post Office Address:

Inventor's Signature Date

Full Name of # 7 joint inventor: Citizenship:

Residence:

Post Office Address:

Inventor's Signature Date

Full Name of # 8 joint inventor: Citizenship:

Residence:

Post Office Address:

Inventor's Signature Date

Package Index

Other Packages

- package .default
- package ecardfile.appl
- package ecardfile.dbappl
- package multiserv.applservlet
- package multiserv.dbmgr
- package multiserv.sessionmgr
- package multiserv.util

package.default

Class Index

- LoginServlet
- SearchServlet

Class LoginServlet

```
Object
   |
   +---GenericServlet
          |
          +---HttpServlet
                 |
                 +---ApplServlet
                        |
                        +---LoginServlet
```

public class **LoginServlet**
extends ApplServlet
Used to override the getApplicationInterface method defined in ApplServlet. This method provides the link between the generic components of package multiserv.applservlet and the application specific functionality.

Constructor Index

- LoginServlet()

Method Index

- getApplicationInterface()
  Overrides the getApplicationInterface method defined in class ApplServlet.
- init(ServletConfig)
  Called when the servlet first gets loaded.

Constructors

- **LoginServlet**

```
public LoginServlet()
```

Methods

- **getApplicationInterface**

```
public multiserv.applservlet.ApplicationInterface
getApplicationInterface()
```

Overrides the getApplicationInterface method defined in class ApplServlet. This method must return an instance of the application specific class which implements ApplicationInterface.
Overrides:
getApplicationInterface in class ApplServlet

● **init**

`public void init(ServletConfig config) throws ServletException`

Called when the servlet first gets loaded. Do our application specific servlet initialization here after calling init() in ApplServlet.

Parameters:

config - servlet configuration information.

Throws: ServletException

a problem occurred during the initialization of the servlet.

Overrides:

init in class ApplServlet

Class SearchServlet

```
Object
   |
   +---GenericServlet
           |
           +---HttpServlet
                   |
                   +---ApplServlet
                           |
                           +---SearchServlet
```

public class **SearchServlet**
extends ApplServlet
Used to override the getApplicationInterface method defined in ApplServlet. This method provides the link between the generic components of package multiserv.applservlet and the application specific functionality.

Constructor Index

- SearchServlet()

Method Index

- getApplicationInterface()
  Overrides the getApplicationInterface method defined in class ApplServlet.
- init(ServletConfig)
  Called when the servlet first gets loaded.

Constructors

- **SearchServlet**

```
public SearchServlet()
```

Methods

- **getApplicationInterface**

```
public multiserv.applservlet.ApplicationInterface
getApplicationInterface()
```

Overrides the getApplicationInterface method defined in class ApplServlet. This method must return an instance of the application specific class which implements ApplicationInterface.
Overrides:

getApplicationInterface in class ApplServlet

- **init**

```
public void init(ServletConfig config) throws ServletException
```

Called when the servlet first gets loaded. Do our application specific servlet initialization here after calling init() in ApplServlet.

Parameters:

config - servlet configuration information.

Throws: ServletException

a problem occurred during the initialization of the servlet.

Overrides:

init in class ApplServlet

package ecardfile.appl

Interface Index

- CommonConfig

Class Index

- CommonApplicationInterface
- EcardNotifier
- LoginApplicationInterface
- SearchApplicationInterface

Interface ecardfile.appl.CommonConfig

public abstract interface **CommonConfig**
Defines the common constants used by the application

Variable Index

- ADDLIST_TAG
- ADDRESSID_TAG
- ADDUSERCONFIRM_TAG
- ADDUSER_TAG
- ALTFIRSTNAME_COL
- ALTFIRSTNAME_TAG
- ALTFIRSTNAME_TOKEN
- BANNER_TOKEN
- BUSINESSCOMMENT_TOKEN
- BUTTON_TAG
- CANCEL_TAG
- CARDEXTRA
- CARDID_TAG
- CARDID_TOKEN
- CATEGORY_TOKEN
- CHANGE
- CHANGEDETAILS_TAG
- CHANGEWHEREAMI_TAG
- COMPANYNAME_COL
- COMPANYNAME_ENC_TOKEN
- COMPANYNAME_TAG
- COMPANYNAME_TOKEN
- CONFIRM_TAG
- CONTACT_COL
- CREATEID_TAG
- DATEOFENTRY_TAG
- DATEOFENTRY_TOKEN
- DELETEUSERCONFIRM_TAG
- DELETEUSER_TAG
- DELETE_TAG
- DISPLAY
- DISPLAYFMT_TAG
- DISPLAYFMT_TOKEN
- DISPLAYLIST_TAG
- DISPLAY_PAGE_TAG

- DOADDWHERE_TAG
- DOCHANGEDETAILS_TAG
- DOCHANGEWHERE_TAG
- DODOWNLOADCARD_TAG
- DODOWNLOADPLIST_TAG
- DODOWNLOADSINGLE_TAG
- DOSEARCH_TAG
- DOUPDATEPLIST_TAG
- DOWNLOADFMT_TAG
- DOWNLOADID_TAG
- DOWNLOADSINGLE_TAG
- DOWNLOAD_TAG
- ECARDID_TAG
- ECARDID_TOKEN
- EDITPRIVACY_TAG
- EMAILAUTH_COL
- EMAILAUTH_TAG
- EMAILAUTH_TOKEN
- EMAILID_TAG
- EPASSWORDCONF_TAG
- EPASSWORD_TAG
- EXPIRYDATE_COL
- FIND_PASSWORD_TAG
- FIRSTNAME_COL
- FIRSTNAME_ENC_TOKEN
- FIRSTNAME_TAG
- FIRSTNAME_TOKEN
- FMT_TAG
- FULLNAME_TOKEN
- HTML
- ID_REWRITE_TOKEN
- ID_TOKEN
- JOBTITLE_TOKEN
- LASTNAME_COL
- LASTNAME_ENC_TOKEN
- LASTNAME_TAG
- LASTNAME_TOKEN
- LC_PRIVACYPREFIX
- LISTITEMS_TOKEN
- LOGIN_SERVLET_TOKEN
- MASK_COL
- MASK_TOKEN
- MIDDLENAME_COL
- MIDDLENAME_TAG
- MIDDLENAME_TOKEN

- MSG_LIST_TOKEN
- NEWUSER_TAG
- OK_TAG
- ONETIME_TAG
- PAGE_TAG
- PAGE_TOKEN
- PASSWORD_COL
- PASSWORD_TOKEN
- PDAPAGE_TAG
- PERSONALLISTID_COL
- PERSONALLISTITEMS_TOKEN
- PHONEID_TAG
- PRIVACYPREFIX
- PVTALTFIRSTNAME_TOKEN
- PVTBUSINESSCOMMENT_TOKEN
- PVTCOMPANYNAME_TOKEN
- PVTCONTACT_COL
- PVTFIRSTNAME_TOKEN
- PVTJOBTITLE_TOKEN
- PVTLASTNAME_TOKEN
- PVTLISTID_COL
- PVTLISTID_TOKEN
- PVTMIDDLENAME_TOKEN
- PVTSUFFIX_TOKEN
- PVTTITLE_TOKEN
- PVTWEBPAGEURL_TOKEN
- ROWID_TAG
- SEARCHID_TAG
- SEARCHLISTITEMS_TOKEN
- SEARCHNAME_TAG
- SEARCH_SERVLET_TOKEN
- SEARCH_TAG
- SITE_ADDRESS_TOKEN
- SOUNDEX_TAG
- SUFFIX_COL
- SUFFIX_TOKEN
- TEMPLATE
- TITLE_COL
- TITLE_TOKEN
- UPDATE_TAG
- USERINFOID_TAG
- USERSID_COL
- USERSID_TOKEN
- WEBPAGEURL_ENC_TOKEN
- WEBPAGEURL_TOKEN

- WHEREAMI
- WHEREAMI_TAG
- WML
- monthNames

  Static array with the names of the months for formatting dates

Variables

- ADDLIST_TAG

```
public static final java.lang.String ADDLIST_TAG
```

- ADDRESSID_TAG

```
public static final java.lang.String ADDRESSID_TAG
```

- ADDUSERCONFIRM_TAG

```
public static final java.lang.String
ADDUSERCONFIRM_TAG
```

- ADDUSER_TAG

```
public static final java.lang.String ADDUSER_TAG
```

- ALTFIRSTNAME_COL

```
public static final java.lang.String ALTFIRSTNAME_COL
```

- ALTFIRSTNAME_TAG

```
public static.final java.lang.String ALTFIRSTNAME_TAG
```

- ALTFIRSTNAME_TOKEN

```
public static final java.lang.String
ALTFIRSTNAME_TOKEN
```

- BANNER_TOKEN

```
public static final java.lang.String BANNER_TOKEN
```

- BUSINESSCOMMENT_TOKEN

```
public static final java.lang.String
BUSINESSCOMMENT_TOKEN
```

- BUTTON_TAG

```
public static final java.lang.String BUTTON_TAG
```

- CANCEL_TAG

```
public static final java.lang.String CANCEL_TAG
```

- CARDEXTRA

public static final short CARDEXTRA
●CARDID_TAG

public static final java.lang.String CARDID_TAG
●CARDID_TOKEN

public static final java.lang.String CARDID_TOKEN
●CATEGORY_TOKEN

public static final java.lang.String CATEGORY_TOKEN
●CHANGE

public static final short CHANGE
●CHANGEDETAILS_TAG

public static final java.lang.String
CHANGEDETAILS_TAG
●CHANGEWHEREAMI_TAG

public static final java.lang.String
CHANGEWHEREAMI_TAG
●COMPANYNAME_COL

public static final java.lang.String COMPANYNAME_COL
●COMPANYNAME_ENC_TOKEN

public static final java.lang.String
COMPANYNAME_ENC_TOKEN
●COMPANYNAME_TAG

public static final java.lang.String COMPANYNAME_TAG
●COMPANYNAME_TOKEN

public static final java.lang.String
COMPANYNAME_TOKEN
●CONFIRM_TAG

public static final java.lang.String CONFIRM_TAG
●CONTACT_COL

public static final java.lang.String CONTACT_COL
●CREATEID_TAG

public static final java.lang.String CREATEID_TAG
●DATEOFENTRY_TAG

public static final java.lang.String DATEOFENTRY_TAG
●DATEOFENTRY_TOKEN

public static final java.lang.String
DATEOFENTRY_TOKEN
●DELETEUSERCONFIRM_TAG

public static final java.lang.String
DELETEUSERCONFIRM_TAG
●DELETEUSER_TAG

public static final java.lang.String DELETEUSER_TAG
●DELETE_TAG

public static final java.lang.String DELETE_TAG
●DISPLAY

public static final short DISPLAY
●DISPLAYFMT_TAG

public static final java.lang.String DISPLAYFMT_TAG
●DISPLAYFMT_TOKEN

public static final java.lang.String DISPLAYFMT_TOKEN
●DISPLAYLIST_TAG

public static final java.lang.String DISPLAYLIST_TAG
●DISPLAY_PAGE_TAG

public static final java.lang.String DISPLAY_PAGE_TAG
●DOADDWHERE_TAG

public static final java.lang.String DOADDWHERE_TAG
●DOCHANGEDETAILS_TAG

public static final java.lang.String
DOCHANGEDETAILS_TAG
●DOCHANGEWHERE_TAG

public static final java.lang.String
DOCHANGEWHERE_TAG
●DODOWNLOADCARD_TAG

public static final java.lang.String
DODOWNLOADCARD_TAG
●DODOWNLOADPLIST_TAG

public static final java.lang.String
DODOWNLOADPLIST_TAG
●DODOWNLOADSINGLE_TAG

public static final java.lang.String
DODOWNLOADSINGLE_TAG
●DOSEARCH_TAG

public static final java.lang.String DOSEARCH_TAG
●DOUPDATEPLIST_TAG

public static final java.lang.String
DOUPDATEPLIST_TAG
●DOWNLOADFMT_TAG

public static final java.lang.String DOWNLOADFMT_TAG
●DOWNLOADID_TAG

public static final java.lang.String DOWNLOADID_TAG
●DOWNLOADSINGLE_TAG

public static final java.lang.String
DOWNLOADSINGLE_TAG
●DOWNLOAD_TAG

public static final java.lang.String DOWNLOAD_TAG
●ECARDID_TAG

public static final java.lang.String ECARDID_TAG
●ECARDID_TOKEN

public static final java.lang.String ECARDID_TOKEN
●EDITPRIVACY_TAG

public static final java.lang.String EDITPRIVACY_TAG
●EMAILAUTH_COL

public static final java.lang.String EMAILAUTH_COL
●EMAILAUTH_TAG

public static final java.lang.String EMAILAUTH_TAG

●EMAILAUTH_TOKEN

```
public static final java.lang.String EMAILAUTH_TOKEN
```

●EMAILID_TAG

```
public static final java.lang.String EMAILID_TAG
```

●EPASSWORDCONF_TAG

```
public static final java.lang.String
EPASSWORDCONF_TAG
```

●EPASSWORD_TAG

```
public static final java.lang.String EPASSWORD_TAG
```

●EXPIRYDATE_COL

```
public static final java.lang.String EXPIRYDATE_COL
```

●FIND_PASSWORD_TAG

```
public static final java.lang.String
FIND_PASSWORD_TAG
```

●FIRSTNAME_COL

```
public static final java.lang.String FIRSTNAME_COL
```

●FIRSTNAME_ENC_TOKEN

```
public static final java.lang.String
FIRSTNAME_ENC_TOKEN
```

●FIRSTNAME_TAG

```
public static final java.lang.String FIRSTNAME_TAG
```

●FIRSTNAME_TOKEN

```
public static final java.lang.String FIRSTNAME_TOKEN
```

●FMT_TAG

```
public static final java.lang.String FMT_TAG
```

●FULLNAME_TOKEN

```
public static final java.lang.String FULLNAME_TOKEN
```

●HTML

```
public static final java.lang.String HTML
```

●ID_REWRITE_TOKEN

```
public static final java.lang.String ID_REWRITE_TOKEN
```

●ID_TOKEN

public static final java.lang.String ID_TOKEN
●JOBTITLE_TOKEN

public static final java.lang.String JOBTITLE_TOKEN
●LASTNAME_COL

public static final java.lang.String LASTNAME_COL
●LASTNAME_ENC_TOKEN

public static final java.lang.String
LASTNAME_ENC_TOKEN
●LASTNAME_TAG

public static final java.lang.String LASTNAME_TAG
●LASTNAME_TOKEN

public static final java.lang.String LASTNAME_TOKEN
●LC_PRIVACYPREFIX

public static final java.lang.String LC_PRIVACYPREFIX
●LISTITEMS_TOKEN

public static final java.lang.String LISTITEMS_TOKEN
●LOGIN_SERVLET_TOKEN

public static final java.lang.String
LOGIN_SERVLET_TOKEN
●MASK_COL

public static final java.lang.String MASK_COL
●MASK_TOKEN

public static final java.lang.String MASK_TOKEN
●MIDDLENAME_COL

public static final java.lang.String MIDDLENAME_COL
●MIDDLENAME_TAG

public static final java.lang.String MIDDLENAME_TAG
●MIDDLENAME_TOKEN

public static final java.lang.String MIDDLENAME_TOKEN
●MSG_LIST_TOKEN

public static final java.lang.String MSG_LIST_TOKEN
- NEWUSER_TAG

public static final java.lang.String NEWUSER_TAG
- OK_TAG

public static final java.lang.String OK_TAG
- ONETIME_TAG

public static final java.lang.String ONETIME_TAG
- PAGE_TAG

public static final java.lang.String PAGE_TAG
- PAGE_TOKEN

public static final java.lang.String PAGE_TOKEN
- PASSWORD_COL

public static final java.lang.String PASSWORD_COL
- PASSWORD_TOKEN

public static final java.lang.String PASSWORD_TOKEN
- PDAPAGE_TAG

public static final java.lang.String PDAPAGE_TAG
- PERSONALLISTID_COL

public static final java.lang.String
PERSONALLISTID_COL
- PERSONALLISTITEMS_TOKEN

public static final java.lang.String
PERSONALLISTITEMS_TOKEN
- PHONEID_TAG

public static final java.lang.String PHONEID_TAG
- PRIVACYPREFIX

public static final java.lang.String PRIVACYPREFIX
- PVTALTFIRSTNAME_TOKEN

public static final java.lang.String
PVTALTFIRSTNAME_TOKEN
- PVTBUSINESSCOMMENT_TOKEN

public static final java.lang.String
PVTBUSINESSCOMMENT_TOKEN

- PVTCOMPANYNAME_TOKEN

public static final java.lang.String
PVTCOMPANYNAME_TOKEN

- PVTCONTACT_COL

public static final java.lang.String PVTCONTACT_COL

- PVTFIRSTNAME_TOKEN

public static final java.lang.String
PVTFIRSTNAME_TOKEN

- PVTJOBTITLE_TOKEN

public static final java.lang.String
PVTJOBTITLE_TOKEN

- PVTLASTNAME_TOKEN

public static final java.lang.String
PVTLASTNAME_TOKEN

- PVTLISTID_COL

public static final java.lang.String PVTLISTID_COL

- PVTLISTID_TOKEN

public static final java.lang.String PVTLISTID_TOKEN

- PVTMIDDLENAME_TOKEN

public static final java.lang.String
PVTMIDDLENAME_TOKEN

- PVTSUFFIX_TOKEN

public static final java.lang.String PVTSUFFIX_TOKEN

- PVTTITLE_TOKEN

public static final java.lang.String PVTTITLE_TOKEN

- PVTWEBPAGEURL_TOKEN

public static final java.lang.String
PVTWEBPAGEURL_TOKEN

- ROWID_TAG

public static final java.lang.String ROWID_TAG

●SEARCHID_TAG

```
public static final java.lang.String SEARCHID_TAG
```

●SEARCHLISTITEMS_TOKEN

```
public static final java.lang.String
SEARCHLISTITEMS_TOKEN
```

●SEARCHNAME_TAG

```
public static final java.lang.String SEARCHNAME_TAG
```

●SEARCH_SERVLET_TOKEN

```
public static final java.lang.String
SEARCH_SERVLET_TOKEN
```

●SEARCH_TAG

```
public static final java.lang.String SEARCH_TAG
```

●SITE_ADDRESS_TOKEN

```
public static final java.lang.String
SITE_ADDRESS_TOKEN
```

●SOUNDEX_TAG

```
public static final java.lang.String SOUNDEX_TAG
```

●SUFFIX_COL

```
public static final java.lang.String SUFFIX_COL
```

●SUFFIX_TOKEN

```
public static final java.lang.String SUFFIX_TOKEN
```

●TEMPLATE

```
public static final java.lang.String TEMPLATE
```

●TITLE_COL

```
public static final java.lang.String TITLE_COL
```

●TITLE_TOKEN

```
public static final java.lang.String TITLE_TOKEN
```

●UPDATE_TAG

```
public static final java.lang.String UPDATE_TAG
```

●USERINFOID_TAG

```
public static final java.lang.String USERINFOID_TAG
```

- USERSID_COL

```
public static final java.lang.String USERSID_COL
```

- USERSID_TOKEN

```
public static final java.lang.String USERSID_TOKEN
```

- WEBPAGEURL_ENC_TOKEN

```
public static final java.lang.String
WEBPAGEURL_ENC_TOKEN
```

- WEBPAGEURL_TOKEN

```
public static final java.lang.String WEBPAGEURL_TOKEN
```

- WHEREAMI

```
public static final short WHEREAMI
```

- WHEREAMI_TAG

```
public static final java.lang.String WHEREAMI_TAG
```

- WML

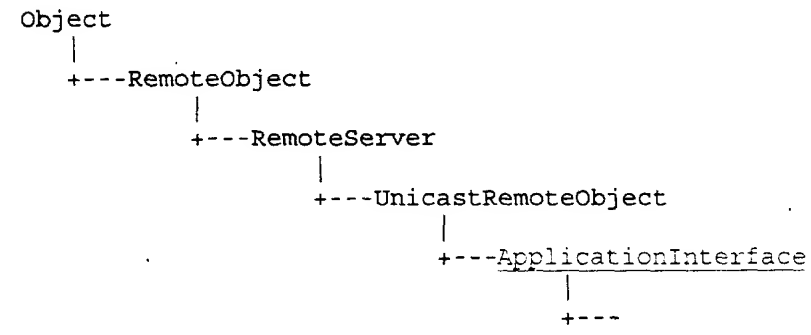
```
public static final java.lang.String WML
```

- monthNames

```
public static final java.lang.String[] monthNames
```

Static array with the names of the months for formatting dates

Class ecardfile.appl.CommonApplicationInterface



public abstract class **CommonApplicationInterface**
extends ApplicationInterface
implements SessionTags, CommonConfig

Version:

$Id: CommonApplicationInterface.java,v 1.44 1999/12/03 02:05:45 peter Exp $

See Also:

ApplicationInterface, SearchApplicationInterface, LoginApplicationInterface, RequestHandler, LoginServlet, SearchServlet

Variable Index

- defaultPdaPage
- verboseErrors

Constructor Index

- ecardfile.appl.CommonApplicationInterface()

Method Index

- accessDenied(String, HttpServletRequest, HttpServletResponse)
  Called when a received request does not contain a valid session id.
- addBannerToken(Hashtable)
  Add the banner text to the token table
- checkAccess(HttpServletRequest)
  Implementation of ApplicationInterface::checkAccess().
- checkPrivacyAccess(DatabaseConnection2, long, long, Hashtable)
- db_error(HttpServletRequest, HttpServletResponse)
  A database error has occurred

- db_error(HttpServletRequest, HttpServletResponse, String)
  A database error has occurred
- destroy()
  Brush teeth before going to bed.
- doc_access_error(HttpServletRequest, HttpServletResponse)
  A document access error has occurred
- getCookie(HttpServletRequest)
  Extract and return our cookie contents from the original HTTP request.
- getCookieTag()
- getFullPath(String)
  Given an HTML document name determine if a full path was specified.
- getOperation(HttpServletRequest)
  Determine the type of operation to be invoked by the request.
- getPassword(HttpServletRequest)
  Get a string representing the user's password
- getServletImgBtnParameter(HttpServletRequest)
- getSessionId(HttpServletRequest)
  Extract and return the session id from the original HTTP request.
- getUserId(HttpServletRequest)
  Get a string representing the user identification
- hiddenField(String, String)
  Return a string of HTML specifying a hidden field.
- init(ApplServlet, String, String)
  Used to initialize the application specific class which implements this interface.
- io_error(HttpServletRequest, HttpServletResponse)
  An io exception has occurred
- isLoggedIn(String, Session)
  Called to check if a user is logged in
- nfe_error(HttpServletRequest, HttpServletResponse)
  A NumberFormatException has occurred
- nfe_error(HttpServletRequest, HttpServletResponse, String)
  A NumberFormatException has occurred
- nse_error(HttpServletRequest, HttpServletResponse)
  A non-serializable exception has occurred
- operationRequiresLogin(String)
  Called to check if a specified operation requires user login.
- re_error(HttpServletRequest, HttpServletResponse)
  A Remote Exception has occurred
- sae_error(HttpServletRequest, HttpServletResponse)
  A Session Access Exception has occurred
- sendDocument(String, HttpServletResponse)
  Send an HTML document to the user replacing the predefined character sequences with their associated values.
- sendError(String, String, String, HttpServletResponse)

- sendError(Hashtable, String, String, String, HttpServletResponse)
- sendError(Hashtable, String, String, HttpServletResponse)
- sendError(HttpServletRequest, String, String, HttpServletResponse)
  General error handling routine.
- sendLoginScreen(HttpServletRequest, HttpServletResponse, String)
- sendLoginScreen(HttpServletRequest, HttpServletResponse, String, Hashtable)
- sendMessage(String, String, Hashtable, String, HttpServletResponse)
- sendMessage(String, String, HttpServletResponse)
- sendMessage(String, Hashtable, String, HttpServletResponse)
- sendParseDocument(Hashtable, String, String, HttpServletResponse)
- sendParseDocument(Hashtable, String, HttpServletResponse)
  Send an HTML document to the user replacing the predefined character sequences with their associated values.
- sendParseTextFile(String, String, Hashtable, String, String, HttpServletResponse)
- sendParseTextFile(String, Hashtable, String, String, HttpServletResponse)
- sendParseTextFile(Hashtable, String, String, HttpServletResponse)
  Send a text file to the user replacing the predefined character sequences with their associated values.
- sendSearchScreen(HttpServletRequest, HttpServletResponse)
- sendSearchScreen(HttpServletRequest, HttpServletResponse, String)
- sendSearchScreen(HttpServletRequest, HttpServletResponse, String, Hashtable)
- sessionFailure(String, HttpServletRequest)
  Implementation of ApplicationInterface:sessionFailure Notify the application that an attempt to access a sesssion using sessionId failed.
- unknownOperation(HttpServletRequest, HttpServletResponse)
  An unknown operation has been requested
- validateSession(String, Session, HttpServletRequest)
  Validate the passed session This method can also be used to implement any security checks or other checks such as if undesirables are accessing the site - such as SPAMMERS
- verboseError(String)
  Method, which displays verbose error, messages to user if debug is turned on

Variables

- **defaultPdaPage**

```
public static final java.lang.String defaultPdaPage
```

- **verboseErrors**

```
protected boolean verboseErrors
```

Constructors

- **CommonApplicationInterface**

```
public CommonApplicationInterface() throws RemoteException
```

Methods

●accessDenied

```
public void accessDenied(String operation,
                         HttpServletRequest req,
                         HttpServletResponse resp)
```

Called when a received request does not contain a valid session id. The application should return an error/warning document to the user. If the error occurred during a normal operation (i.e. other than Logging in or Logging out) it was more than likely due to the session having expired. In this case just bring up the login screen.

Parameters:

operation - The operation which was being attempted.
req - The original HTTP request.
resp - The HTTP response

Overrides:

accessDenied in class ApplicationInterface

●addBannerToken

```
protected void addBannerToken(Hashtable tokens)
```

Add the banner text to the token table

Parameters:

tokens - the token table

●checkAccess

```
public boolean checkAccess(HttpServletRequest req)
```

Implementation of ApplicationInterface::checkAccess(). Check the HTTP request data and decide if access should be allowed. We check to see if the IP Address is locked out.

Parameters:

req - The original HTTP request data.

Returns:

An indication if access is to be granted

Overrides:

checkAccess in class ApplicationInterface

●checkPrivacyAccess

```
public short checkPrivacyAccess(DatabaseConnection2 jdbc,
                                long userId,
                                long cardId,
                                Hashtable cardRow)
```

●db_error

```
public void db_error(HttpServletRequest req,
                     HttpServletResponse resp)
```

A database error has occurred

Overrides:

db_error in class ApplicationInterface

- **db_error**

```
public void db_error(HttpServletRequest req,
                     HttpServletResponse resp,
                     String msg)
```

A database error has occurred

Overrides:

db_error in class ApplicationInterface

- **destroy**

```
protected void destroy()
```

Brush teeth before going to bed.

Overrides:

destroy in class ApplicationInterface

- **doc_access_error**

```
public void doc_access_error(HttpServletRequest req,
                             HttpServletResponse resp)
```

A document access error has occurred

Overrides:

doc_access_error in class ApplicationInterface

- **getCookie**

```
public java.lang.String getCookie(HttpServletRequest req)
```

Extract and return our cookie contents from the original HTTP request. The cookie used by this application holds the session id.

Parameters:

req - the original HTTP request.

Returns:

the session id associated with the request. If no session id is found returns null.

- **getCookieTag**

```
public java.lang.String getCookieTag()
```

- **getFullPath**

```
protected java.lang.String getFullPath(String document)
```

Given an HTML document name determine if a full path was specified. If not tack getProperty("ecardfile.html.base") on to the front of the document

Parameters:

document - the filename of the HTML document to be sent

Returns:

full URL of document

- **getOperation**

```
public java.lang.String getOperation(HttpServletRequest req)
```

Determine the type of operation to be invoked by the request. Note that session creation and session destruction requests are indicated by the operation strings defined by RequestHandler.CREATE and RequestHandler.DESTROY respectively.

Parameters:

req - The HTTP request.

Returns:

A string describing the operation to carry out.

Overrides:

getOperation in class ApplicationInterface

- **getPassword**

```
public java.lang.String getPassword(HttpServletRequest req)
```

Get a string representing the user's password

Parameters:

req - The original HTTP request data.

Returns:

A string containing the password or null if the password tag isn't found in the HTTP request.

Overrides:

getPassword in class ApplicationInterface

- **getServletImgBtnParameter**

```
public java.lang.String
getServletImgBtnParameter(HttpServletRequest req) throws
ServletException, IOException
```

- **getSessionId**

```
public java.lang.String getSessionId(HttpServletRequest req)
throws ServletException, IOException
```

Extract and return the session id from the original HTTP request.

Parameters:

req - the original HTTP request.

Returns:

the session id associated with the request. If no session id is found returns null.

Throws: ServletException

Throws: IOException

Overrides:

getSessionId in class ApplicationInterface

- **getUserId**

```
public java.lang.String getUserId(HttpServletRequest req)
```

Get a string representing the user identification

Parameters:
req - The original HTTP request data.
Returns:
A string containing the user id or null if the user id tag isn't found in the HTTP request.
Overrides:
getUserId in class ApplicationInterface

- **hiddenField**

```
protected java.lang.String hiddenField(String name,
                                       String value)
```

Return a string of HTML specifying a hidden field.
Parameters:
name - NAME of the hidden field
value - VALUE of the hidden field
Returns:
HTML string as described above.

- **init**

```
public void init(ApplServlet servlet,
                 String managerName,
                 String rmiHost) throws ServletException,
IOException
```

Used to initialize the application specific class, which implements this interface. The servlet configuration is passed in so that the servlet environment is available to the application code.
Parameters:
applServlet - The Servlet instance which owns this ApplicationInterface instance.
Throws: ServletException
Overrides:
init in class ApplicationInterface

- **io_error**

```
public void io_error(HttpServletRequest req,
                     HttpServletResponse resp)
```

An io exception has occurred
Overrides:
io_error in class ApplicationInterface

- **isLoggedIn**

```
protected boolean isLoggedIn(String sessionId,
                             Session session) throws
SessionAccessException, IOException
```

Called to check if a user is logged in
Parameters:
sessionId - The Id of the current session

session - The current session

Returns:

True if the user is currently logged in

- **nfe_error**

```
public void nfe_error(HttpServletRequest req,
                      HttpServletResponse resp)
```

A NumberFormatException has occurred

Overrides:

nfe_error in class ApplicationInterface

- **nfe_error**

```
public void nfe_error(HttpServletRequest req,
                      HttpServletResponse resp,
                      String msg)
```

A NumberFormatException has occurred

Overrides:

nfe_error in class ApplicationInterface

- **nse_error**

```
public void nse_error(HttpServletRequest req,
                      HttpServletResponse resp)
```

A non-serializable exception has occurred

Overrides:

nse_error in class ApplicationInterface

- **operationRequiresLogin**

```
protected boolean operationRequiresLogin(String operation)
```

Called to check if a specified operation requires user login.

Parameters:

operation - The operation which was being attempted.

Returns:

True if the operation requires login, false otherwise

- **re_error**

```
public void re_error(HttpServletRequest req,
                     HttpServletResponse resp)
```

A Remote Exception has occurred

Overrides:

re_error in class ApplicationInterface

- **sae_error**

```
public void sae_error(HttpServletRequest req,
                      HttpServletResponse resp)
```

A Session Access Exception has occurred

Overrides:

sae_error in class ApplicationInterface

- **sendDocument**

```
public void sendDocument(String document,
                         HttpServletResponse resp)
```

Send an HTML document to the user replacing the predefined character sequences with their associated values.

Parameters:

document - the file name of the HTML document to be sent. All documents are accessed relative to our ecardfile.html.base property

resp - provides methods to respond to the original HTTP request.

- **sendError**

```
public void sendError(String format,
                      String msg1,
                      String msg2,
                      HttpServletResponse resp)
```

- **sendError**

```
public void sendError(Hashtable tokens,
                      String format,
                      String msg1,
                      String msg2,
                      HttpServletResponse resp)
```

- **sendError**

```
public void sendError(Hashtable tokens,
                      String msg1,
                      String msg2,
                      HttpServletResponse resp)
```

- **sendError**

```
public void sendError(HttpServletRequest req,
                      String msg1,
                      String msg2,
                      HttpServletResponse resp)
```

General error handling routine.

Overrides:

sendError in class ApplicationInterface

- **sendLoginScreen**

```
protected void sendLoginScreen(HttpServletRequest req,
                               HttpServletResponse resp,
                               String format) throws IOException
```

- **sendLoginScreen**

```
protected void sendLoginScreen(HttpServletRequest req,
                               HttpServletResponse resp,
                               String format,
                               Hashtable initTokens) throws
IOException
```

- **sendMessage**

```
public void sendMessage(String pdaPage,
                        String MessageFile,
                        Hashtable tokens,
                        String format,
                        HttpServletResponse resp)
```

●**sendMessage**

```
public void sendMessage(String MessageFile,
                        String format,
                        HttpServletResponse resp)
```

●**sendMessage**

```
public void sendMessage(String MessageFile,
                        Hashtable tokens,
                        String format,
                        HttpServletResponse resp)
```

●**sendParseDocument**

```
public void sendParseDocument(Hashtable tokenTable,
                              String document,
                              String format,
                              HttpServletResponse resp)
```

●**sendParseDocument**

```
public void sendParseDocument(Hashtable tokenTable,
                              String document,
                              HttpServletResponse resp)
```

Send an HTML document to the user replacing the predefined character sequences with their associated values.

Parameters:

tokenTable - the tokens and values to be replaced in the document

document - the file name of the HTML document to be sent. All documents are accessed relative to our ecardfile.html.base property session.

resp - provides methods to respond to the original HTTP request.

●**sendParseTextFile**

```
public void sendParseTextFile(String contentType,
                              String fileName,
                              Hashtable tokenTable,
                              String document,
                              String format,
                              HttpServletResponse resp)
```

●**sendParseTextFile**

```
public void sendParseTextFile(String contentType,
                              Hashtable tokenTable,
                              String document,
                              String format,
                              HttpServletResponse resp)
```

● sendParseTextFile

```
public void sendParseTextFile(Hashtable tokenTable,
                              String document,
                              String format,
                              HttpServletResponse resp)
```

Send a text file to the user replacing the predefined character sequences with their associated values.

Parameters:

contentType - The content type of the text file
tokenTable - the tokens and values to be replaced in the document
document - the file name of the HTML document to be sent. All documents are accessed relative to our ecardfile.html.base property session.
format - the type of file to send, html or wml
resp - provides methods to respond to the original HTTP request.

● sendSearchScreen

```
protected void sendSearchScreen(HttpServletRequest req,
                                HttpServletResponse resp)
```

● sendSearchScreen

```
protected void sendSearchScreen(HttpServletRequest req,
                                HttpServletResponse resp,
                                String format) throws
IOException
```

● sendSearchScreen

```
protected void sendSearchScreen(HttpServletRequest req,
                                HttpServletResponse resp,
                                String format,
                                Hashtable initTokens) throws
IOException
```

● sessionFailure

```
public boolean sessionFailure(String sessionId,
                              HttpServletRequest req)
```

Implementation of ApplicationInterface:sessionFailure Notify the application that an attempt to access a session using sessionId failed. This could be due to a bogus sessionId or an expired session. We keep track of the failures per IP Address so that they can be checked in checkAccess()

Parameters:

req - The original HTTP request data.
sessionId - The session id string for the current session.

Returns:

If true a new session will be created

Overrides:

sessionFailure in class ApplicationInterface

- **unknownOperation**

```
public void unknownOperation(HttpServletRequest req,
                             HttpServletResponse resp)
```

An unknown operation has been requested

Overrides:

unknownOperation in class ApplicationInterface

- **validateSession**

```
public boolean validateSession(String sessionId,
                               Session session,
                               HttpServletRequest req) throws
Exception
```

Validate the passed session This method can also be used to implement any security checks or other checks such as if undesirables are accessing the site - such as SPAMMERS

Parameters:

session - The session to validate.

req - The original HTTP request data.

Returns:

An indication if the session is valid

Overrides:

validateSession in class ApplicationInterface

- **verboseError**

```
protected java.lang.String verboseError(String msg)
```

Method which displays verbose error messages to user if debug is turned on

Class ecardfile.appl.EcardNotifier

```
Object
   |
   +---ecardfile.appl.EcardNotifier
```

public class **EcardNotifier**
extends Object
implements Runnable

Constructor Index

ecardfile.appl.EcardNotifier(CommonApplicationInterface, String, long, String)

Method Index

- interrupt()
- run()
- start()
- stop()

Constructors

EcardNotifier

```
public EcardNotifier(CommonApplicationInterface parent,
                     String templateCache,
                     long cardId,
                     String eCardId)
```

Methods

interrupt

```
public void interrupt()
```

run

```
public void run()
```

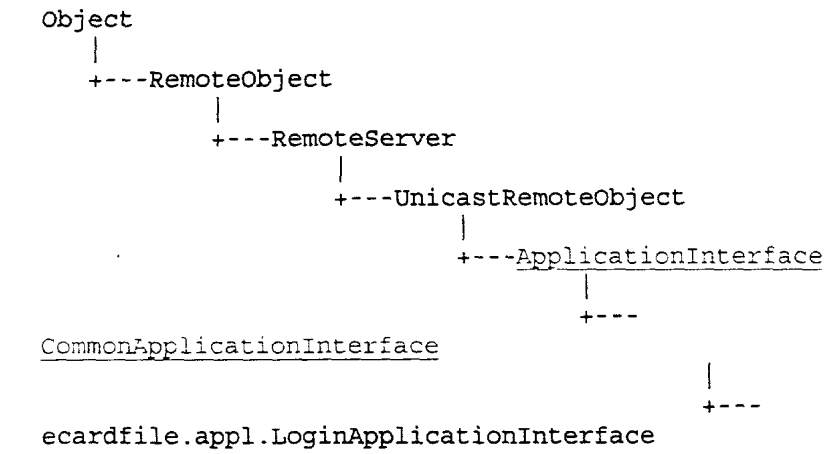
start

```
public void start()
```

stop

```
public void stop()
```

Class ecardfile.appl.LoginApplicationInterface



public class **LoginApplicationInterface**
extends CommonApplicationInterface
implements CommonConfig
The class implements the application behavior for the LoginServlet. Many of the methods are hooks called by RequestHandler instances invoked by LoginServlet (i.e. ApplServlet).

Version:

$Id: LoginApplicationInterface.java,v 1.23 1999/11/24 02:24:41 peter Exp $

See Also:

RequestHandler, LoginServlet, CommonConfig, SessionImpl

Constructor Index

Method Index

- init(ApplServlet, String, String)
  Used to initialize the application specific class which implements this interface.
- notify(SessionNotification)
  Receives (asynchronous???) notifications from XXX This overrides the corresponding method in ApplicationInterface which in turn implements the method
- postAutheniticate(String, Session, HttpServletRequest, HttpServletResponse)
  Post authentication functionality.
- postDestroy(HttpServletRequest, HttpServletResponse)
  Called by the RequestHandler immediately after the destruction of the session object.
- preDestroy(String, Session, HttpServletRequest, HttpServletResponse)
  Called by RequestHandler prior to the destruction of the session object.

Constructors

LoginApplicationInterface

```
public LoginApplicationInterface() throws RemoteException
```

Methods

accessDenied

```
public void accessDenied(String operation,
                         HttpServletRequest req,
                         HttpServletResponse resp)
```

Called when a received request does not contain a valid session id. The application should return an error/warning document to the user. If the error occurred during a normal operation (i.e. other than Logging in or Logging out) it was more than likely due to the session having expired. In this case just bring up the login screen.

Parameters:

operation - The operation which was being attempted.
req - The original HTTP request.
resp - The HTTP response

Overrides:

accessDenied in class CommonApplicationInterface

authenticateUser

```
public multiserv.sessionmgr.GenericSession
authenticateUser(String userName,
String password,
HttpServletRequest req) throws RemoteException,
SessionAccessException, NotSerializableException,
ServletException, IOException
```

Authenticate the user using a user id and password combination. Authentication is done against the database. This method can also be used to implement any pre-session creation functionality.

Parameters:

userName - The user name string.
password - The user's password string.
req - The original HTTP request data.

Returns:

A session object containing any initial session data. The returned object must be a sub-class of GenericSession. This will be used to initialize the session object in the session manager. If the authentication fails, null is returned.

Throws: RemoteException

A problem occured while trying to create the session object in the SessionManager

Overrides:

authenticateUser in class ApplicationInterface

See Also:

UserAuth, SessionImpl

- **executeOperation**

```
public void executeOperation(String operation,
                             String sessionId,
                             Session session,
                             HttpServletRequest req,
                             HttpServletResponse resp)
```

Called by RequestHandler to execute an application defined operation. Valid operations are:

Arthroscopy
Performed on Peter's knee

Operations can be specification of the OP_TAG as a hidden field or as part of a Query String in a GET request. Alternatively, certain FORM buttons are defined to execute specific operations.

Parameters:

operation - Specifies the operation to perform. This should be one of the operation strings returned by getOperation.
session - An interface to the current session object.
req - The data from the original HTTP request.
resp - Provides methods for responding to the request.

Overrides:

executeOperation in class ApplicationInterface

- **getCookieTag**

```
public java.lang.String getCookieTag()
```

Return the application specific cookie tag

Returns:

name of the application specific cookie

Overrides:

getCookieTag in class CommonApplicationInterface

- **init**

```
public void init(ApplServlet servlet,
                 String managerName,
                 String rmiHost) throws ServletException,
IOException
```

Used to initialize the application specific class which implements this interface. The servlet configuration is passed in so that the servlet environment is available to the application code.

Parameters:

applServlet - The Servlet instance which owns this ApplicationInterface instance.

Throws: ServletException

Overrides:

init in class CommonApplicationInterface

- **notify**

```
public void notify(SessionNotification nofn)
```

Receives (asynchronous???) notifications from XXX This overrides the corresponding method in ApplicationInterface which in turn implements the method

Parameters:

nofn - An interface to the object carrying notification information.

Overrides:

notify in class ApplicationInterface

- **postAuthenticate**

```
public void postAuthenticate(String sessionId,
                             Session session,
                             HttpServletRequest req,
                             HttpServletResponse resp) throws
IOException, SessionAccessException, ServletException,
IOException
```

Post authentication functionality. If the authentication had failed a screen displaying an appropraite message is displayed.

Overrides:

postAuthenticate in class ApplicationInterface

- **postDestroy**

```
public void postDestroy(HttpServletRequest req,
                        HttpServletResponse resp)
```

Called by the RequestHandler immediately after the destruction of the session object. The session is finished. Display the Login screen.

Parameters:

req - The data from the original HTTP request.

resp - Provides methods for responding to the request.

Overrides:

postDestroy in class ApplicationInterface

- **preDestroy**

```
public void preDestroy(String sessionId,
                       Session session,
                       HttpServletRequest req,
                       HttpServletResponse resp)
```

Called by RequestHandler prior to the destruction of the session object.

Parameters:

session - An interface to the associated session object.

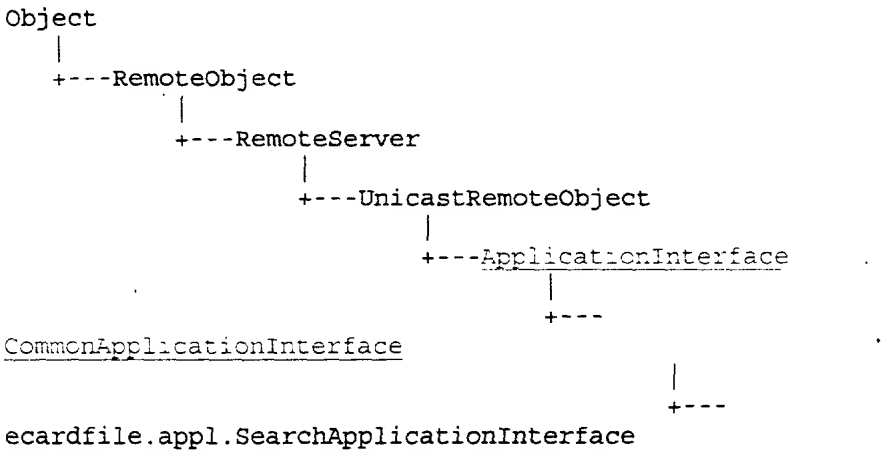
req - The data from the original HTTP request.

resp - Provides methods for responding to the request.

Overrides:

preDestroy in class ApplicationInterface

Class ecardfile.appl.SearchApplicationInterface



public class **SearchApplicationInterface**
extends CommonApplicationInterface
implements CommonConfig
The class implements the application behavior for the SearchServlet. Many of the methods are hooks called by RequestHandler instances invoked by SearchServlet (i.e. ApplServlet).

Version:

$Id: SearchApplicationInterface.java,v 1.59 1999/12/03 01:10:09 peter Exp $

See Also:

RequestHandler, SearchServlet, CommonConfig, SessionImpl

Variable Index

- PRIVATE_ACCESS
- PUBLIC_ACCESS

Constructor Index

- ecardfile.appl.SearchApplicationInterface()

Method Index

- accessDenied(String, HttpServletRequest, HttpServletResponse)
  Called when a received request does not contain a valid session id.
- authenticateUser(String, String, HttpServletRequest)
  Authenticate the user using a user id and password combination.
- checkPrivacyAccess(short, Hashtable)

Checks all values in the card that is about to be displayed, if the privacy value for any value is higher than that granted by the card's owner the element is removed from the hashtable and the value is not displayed.

- destroy()
  Brush teeth before going to bed.
- executeOperation(String, String, Session, HttpServletRequest, HttpServletResponse)
  Called by RequestHandler to execute an application defined operation.
- getCookieTag()
  Return the application specific cookie tag
- getPrivacyAccess(DatabaseConnection2, long, long)
  Get the privacy mask for the logged in user and the card being displayed
- getPrivacyAccess(DatabaseConnection2, long, long, Hashtable)
- init(ApplServlet, String, String)
  Used to initialize the application specific class, which implements this interface.
- notify(SessionNotification)
  Receives (asynchronous???) notifications from XXX This overrides the corresponding method in ApplicationInterface which in turn implements the method
- postAuthenticate(String, Session, HttpServletRequest, HttpServletResponse)
  Post authentication functionality.
- postDestroy(HttpServletRequest, HttpServletResponse)
  Called by the RequestHandler immediately after the destruction of the session object.
- preDestroy(String, Session, HttpServletRequest, HttpServletResponse)
  Called by RequestHandler prior to the destruction of the session object.

Variables

- **PRIVATE_ACCESS**

```
public static final short PRIVATE_ACCESS
```

- **PUBLIC_ACCESS**

```
public static final short PUBLIC_ACCESS
```

Constructors

- **SearchApplicationInterface**

```
public SearchApplicationInterface() throws RemoteException
```

Methods

- **accessDenied**

```
public void accessDenied(String operation,
                         HttpServletRequest req,
```

```
                                HttpServletResponse resp)
```

Called when a received request does not contain a valid session id. The application should return an error/warning document to the user. If the error occured during a normal operation (i.e. other than Logging in or Logging out) it was more than likely due to the session having expired. In this case just bring up the login screen.

Parameters:

operation - The operation which was being attempted.

req - The original HTTP request.

resp - The HTTP response

Overrides:

accessDenied in class CommonApplicationInterface

- **authenticateUser**

```
public multiserv.sessionmgr.GenericSession
authenticateUser(String userName,

String password,

HttpServletRequest req) throws RemoteException,
SessionAccessException, NotSerializableException,
ServletException, IOException
```

Authenticate the user using a user id and password combination. Authentication is done against the database. This method can also be used to implement any pre-session creation functionality.

Parameters:

userName - The user name string.

password - The user's password string.

req - The original HTTP request data.

Returns:

A session object containing any initial session data. The returned object must be a sub-class of GenericSession. This will be used to initialize the session object in the session manager. If the authentication fails, null is returned.

Throws: RemoteException

A problem occurred while trying to create the session object in the SessionManager

Overrides:

authenticateUser in class ApplicationInterface

See Also:

UserAuth, SessionImpl

- **checkPrivacyAccess**

```
public void checkPrivacyAccess(short privacyLevel,
                               Hashtable cardRow)
```

Checks all values in the card that is about to be displayed, if the privacy value for any value is higher than that granted by the card's owner the element is removed from the hashtable and the value is not displayed.

Parameters:

privacyLevel - Privacy value

cardRow - Hashtable from which the elements are removed

- **destroy**

```
protected void destroy()
```

Brush teeth before going to bed.

Overrides:

destroy in class CommonApplicationInterface

- **executeOperation**

```
public void executeOperation(String operation,
                             String sessionId,
                             Session session,
                             HttpServletRequest req,
                             HttpServletResponse resp)
```

Called by RequestHandler to execute an application defined operation.

Valid operations are:

Arthroscopy

Performed on Peter's knee

Operations can be specification of the OP_TAG as a hidden field or as part of a Query String in a GET request. Alternatively, certain FORM buttons are defined to execute specific operations.

Parameters:

operation - Specifies the operation to perform. This should be one of the operation strings returned by getOperation.

session - An interface to the current session object.

req - The data from the original HTTP request.

resp - Provides methods for responding to the request.

Overrides:

executeOperation in class ApplicationInterface

- **getCookieTag**

```
public java.lang.String getCookieTag()
```

Return the application specific cookie tag

Returns:

name of the application specific cookie

Overrides:

getCookieTag in class CommonApplicationInterface

- **getPrivacyAccess**

```
public short getPrivacyAccess(DatabaseConnection2 jdbc,
                              long loginId,
                              long cardId)
```

Get the privacy mask for the logged in user and the card being displayed

Parameters:

jdbc - An open database connection

loginId - user id of currently logged in user (0 if not logged in)

cardId - user id of the card details being checked

card - Card to add mask and privacy row id if required

- **getPrivacyAccess**

```
public void getPrivacyAccess(DatabaseConnection jdbc,
                             long loginId,
                             long cardId,
                             Hashtable card)
```

- init

```
public void init(ApplServlet servlet,
                 String managerName,
                 String rmiHost) throws ServletException,
IOException
```

Used to initialize the application specific class, which implements this interface. The servlet configuration is passed in so that the servlet environment is available to the application code.

Parameters:

applServlet - The Servlet instance which owns this ApplicationInterface instance.

Throws: ServletException

Overrides:

init in class CommonApplicationInterface

- **notify**

```
public void notify(SessionNotification nofn)
```

Receives (asynchronous???) notifications from XXX This overrides the corresponding method in ApplicationInterface which in turn implements the method

Parameters:

nofn - An interface to the object carrying notification information.

Overrides:

notify in class ApplicationInterface

- **postAuthenticate**

```
public void postAuthenticate(String sessionId,
                             Session session,
                             HttpServletRequest req,
                             HttpServletResponse resp) throws
IOException, SessionAccessException, ServletException
```

Post authentication functionality. If the authentication had failed a screen displaying an appropriate message is displayed.

Overrides:

postAuthenticate in class ApplicationInterface

● postDestroy

```
public void postDestroy(HttpServletRequest req,
                        HttpServletResponse resp)
```

Called by the RequestHandler immediately after the destruction of the session object. The session is finished. Display the Login screen.

Parameters:

req - The data from the original HTTP request.

resp - Provides methods for responding to the request.

Overrides:

postDestroy in class ApplicationInterface

● preDestroy

```
public void preDestroy(String sessionId,
                       Session session,
                       HttpServletRequest req,
                       HttpServletResponse resp)
```

Called by RequestHandler prior to the destruction of the session object.

Parameters:

session - An interface to the associated session object.

req - The data from the original HTTP request.

resp - Provides methods for responding to the request.

Overrides:

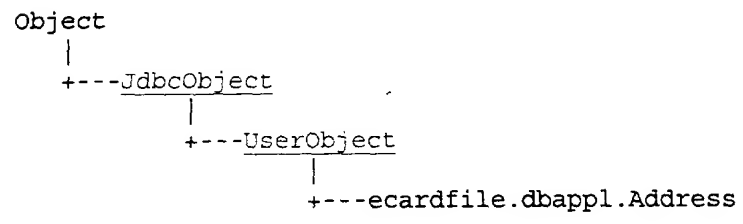
preDestroy in class ApplicationInterface

package ecardfile.dbappl

Class Index

- Address
- Banner
- BannerCache
- DatabaseConnection2
- Email
- InactiveAddress
- InactiveDatabaseConnection
- InactiveEmail
- InactivePhone
- InactiveUser
- LockedIP
- Lookup
- LookupCache
- PersonalList
- Phone
- PrivateList
- User
- UserInfo
- UserObject
- WhereAmI

Class ecardfile.dbappl.Address



public class **Address**
extends UserObject

Constructor Index

- ecardfile.dbappl.Address(Connection)

Constructors

- **Address**

```
public Address(Connection connect) throws JdbcException
```

Class ecardfile.dbappl.Banner

```
Object
   |
   +---JdbcObject
          |
          +---ecardfile.dbappl.Banner
```

public class **Banner**
extends JdbcObject

Constructor Index

- ecardfile.dbappl.Banner(Connection)

Constructors

- **Banner**

```
public Banner(Connection connect) throws JdbcException
```

Class ecardfile.dbappl.BannerCache

```
Object
   |
   +---Cache
          |
          +---TableCache
                 |
                 +---ecardfile.dbappl.BannerCache
```

public class **BannerCache**
extends TableCache
Cache of the Banner table
See Also:
TableCache

Constructor Index

- ecardfile.dbappl.BannerCache()
  Shouldn't use this constructor
- ecardfile.dbappl.BannerCache(JdbcConnectionBroker2, long)

Method Index

- getJdbcObject(Connection)
  This method should be defined in the sub class.
- getRandomAd()
  Get a random Banner ad.
- populate()
  The database table is populated by using a JDBC object created by getJdbcObject.

Constructors

BannerCache

```
public BannerCache()
```

Shouldn't use this constructor

BannerCache

```
public BannerCache(JdbcConnectionBroker2 connMgr,
                   long secs) throws IOException
```

Parameters:
connMgr - - The JDBC Connection manager
secs - - The cache is repopulated every *secs* seconds

Methods

●getJdbcObject

```
protected multiserv.dbmgr.JdbcObject getJdbcObject(Connection
connect) throws JdbcException
```

This method should be defined in the sub class.

Overrides:

getJdbcObject in class TableCache

●getRandomAd

```
public java.lang.String getRandomAd()
```

Get a random Banner ad. Basically it just returns an HTML String by using a random key into the banner table cache.

Returns:

String of HTML which can display an ad banner

See Also:

populate

●populate

```
public void populate()
```

The database table is populated by using a JDBC object created by getJdbcObject. We key a vector of keys locally. As records get added/deleted from the table the set of Ids will change. To retrieve random ads we randomly index into out local set of keys and use that key to pull an add from the cache.

Overrides:

populate in class TableCache

Class ecardfile.dbappl.DatabaseConnection2

```
Object
   |
   +---JdbcConnection
          |
          +---ecardfile.dbappl.DatabaseConnection2
```

public class **DatabaseConnection2**
extends JdbcConnection

Variable Index

- DEBUG

Constructor Index

- ecardfile.dbappl.DatabaseConnection2()
  Only to be used for JdbcConnectionBroker/JdbcConnectionFactory
- ecardfile.dbappl.DatabaseConnection2(String, String, String)

Method Index

- Address()
- Banner()
- Email()
- Initialize()
- LockedIP()
- Lookup()
- PersonalList()
- Phone()
- PrivateList()
- User()
- WhereAmI()
- close()
- getInstance(String, String, String)
- main(String[])

Variables

● DEBUG

```
public static boolean DEBUG
```

Constructors

● DatabaseConnection2

```
public DatabaseConnection2()
```
Only to be used for JdbcConnectionBroker/JdbcConnectionFactory

● DatabaseConnection2

```
public DatabaseConnection2(String URL,
                           String username,
                           String password) throws JdbcException
```

Methods

● Address

```
public ecardfile.dbappl.Address Address() throws JdbcException
```

● Banner

```
public ecardfile.dbappl.Banner Banner() throws JdbcException
```

● Email

```
public ecardfile.dbappl.Email Email() throws JdbcException
```

● Initialize

```
public void Initialize() throws JdbcException
```

Overrides:

Initialize in class JdbcConnection

● LockedIP

```
public ecardfile.dbappl.LockedIP LockedIP() throws JdbcException
```

● Lookup

```
public ecardfile.dbappl.Lookup Lookup() throws JdbcException
```

● PersonalList

```
public ecardfile.dbappl.PersonalList PersonalList() throws
JdbcException
```

● Phone

```
public ecardfile.dbappl.Phone Phone() throws JdbcException
```

● PrivateList

```
public ecardfile.dbappl.PrivateList PrivateList() throws
JdbcException
```

● User

```
public ecardfile.dbappl.User User() throws JdbcException
```

● WhereAmI

```
public ecardfile.dbappl.WhereAmI WhereAmI() throws JdbcException
```

● close

```
public void close() throws SQLException
```

Overrides:

close in class JdbcConnection

- **getInstance**

```
public multiserv.dbmgr.JdbcConnection getInstance(String URL,
                                                  String
username,
                                                  String
password) throws JdbcException
```

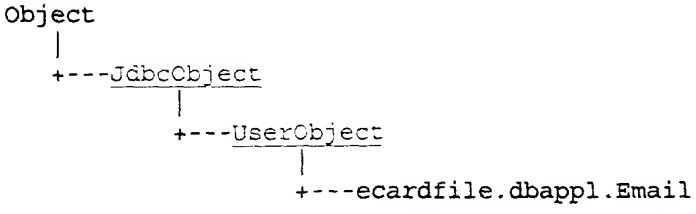
Overrides:

getInstance in class JdbcConnection

- **main**

```
public static void main(String[] args)
```

Class ecardfile.dbappl.Email



public class **Email**
extends UserObject

Constructor Index

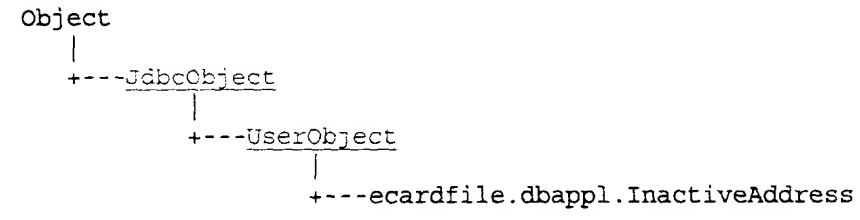
- ecardfile.dbappl.Email(Connection)

Constructors

- **Email**

```
public Email(Connection connect) throws JdbcException
```

Class ecardfile.dbappl.InactiveAddress



public class **InactiveAddress**
extends UserObject

Constructor Index

ecardfile.dbappl.InactiveAddress(Connection)

Constructors

InactiveAddress

```
public InactiveAddress(Connection connect) throws JdbcException
```

Class ecardfile.dbappl.InactiveDatabaseConnection

```
Object
   |
   +---ecardfile.dbappl.InactiveDatabaseConnection
```

public class **InactiveDatabaseConnection**
extends Object

Variable Index

- DEBUG

Constructor Index

- ecardfile.dbappl.InactiveDatabaseConnection(Connection)

Method Index

- InactiveAddress()
- InactiveEmail()
- InactivePhone()
- InactiveUser()

Variables

DEBUG

```
public static boolean DEBUG
```

Constructors

InactiveDatabaseConnection

```
public InactiveDatabaseConnection(Connection conn)
```

Methods

InactiveAddress

```
public ecardfile.dbappl.InactiveAddress InactiveAddress() throws
JdbcException
```

InactiveEmail

```
public ecardfile.dbappl.InactiveEmail InactiveEmail() throws
JdbcException
```

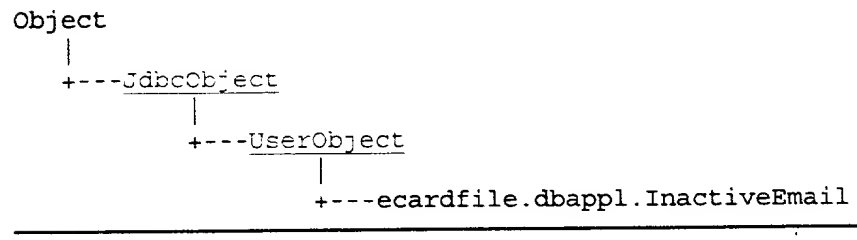
InactivePhone

```
public ecardfile.dbappl.InactivePhone InactivePhone() throws
JdbcException
```

- **InactiveUser**

```
public ecardfile.dbappl.InactiveUser InactiveUser() throws
JdbcException
```

Class ecardfile.dbappl.InactiveEmail



public class **InactiveEmail**
extends UserObject

Constructor Index

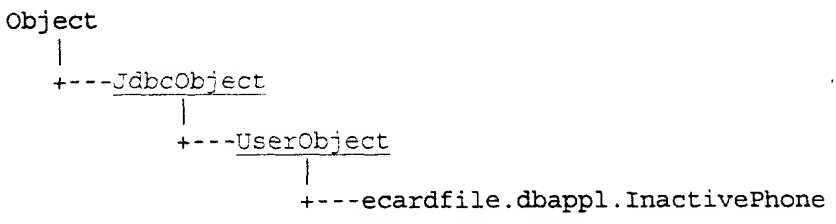
ecardfile.dbappl.InactiveEmail(Connection)

Constructors

InactiveEmail

```
public InactiveEmail(Connection connect) throws JdbcException
```

Class ecardfile.dbappl.InactivePhone



public class **InactivePhone**
extends UserObject

Constructor Index

- ecardfile.dbappl.InactivePhone(Connection)

Constructors

- **InactivePhone**

```
public InactivePhone(Connection connect) throws JdbcException
```

Class ecardfile.dbappl.InactiveUser

```
Object
   |
   +---JdbcObject
          |
          +---ecardfile.dbappl.InactiveUser
```

public class **InactiveUser**
extends JdbcObject

Constructor Index

- ecardfile.dbappl.InactiveUser(Connection)

Method Index

- Delete(InactiveDatabaseConnection, long)
- Get(String)
- Get(String, String)
- Insert(InactiveDatabaseConnection, String[], Vector, Vector, Vector)
- Update(InactiveDatabaseConnection, String[], Vector, Vector, Vector)

Constructors

InactiveUser

```
public InactiveUser(Connection connect) throws JdbcException
```

Methods

Delete

```
public int Delete(InactiveDatabaseConnection database,
                  long userId) throws JdbcException
```

Get

```
public java.util.Hashtable Get(String szECardId) throws
JdbcException
```

Get

```
public java.util.Hashtable Get(String szECardId,
                               String szSessionId) throws
JdbcException
```

Insert

```
public long Insert(InactiveDatabaseConnection database,
                   String[] userRow,
```

```
                    Vector addressRows,
                    Vector phoneRows,
                    Vector emailRows) throws JdbcException
```

- **Update**

```
public long Update(InactiveDatabaseConnection database,
                    String[] userRow,
                    Vector addressRows,
                    Vector phoneRows,
                    Vector emailRows) throws JdbcException
```

Class ecardfile.dbappl.LockedIP

```
Object
    |
    +---JdbcObject
            |
            +---ecardfile.dbappl.LockedIP
```

public class **LockedIP**
extends JdbcObject

Variable Index

- psAll

Constructor Index

- ecardfile.dbappl.LockedIP(Connection)

Method Index

- QueryAll()

Variables

- **psAll**

```
protected java.sql.PreparedStatement psAll
```

Constructors

- **LockedIP**

```
public LockedIP(Connection connect) throws JdbcException
```

Methods

- **QueryAll**

```
public java.util.Vector QueryAll() throws JdbcException
```

Overrides:
QueryAll in class JdbcObject

Class ecardfile.dbappl.Lookup

```
Object
   |
   +---JdbcObject
          |
          +---ecardfile.dbappl.Lookup
```

public class **Lookup**
extends JdbcObject

Constructor Index

ecardfile.dbappl.Lookup(Connection)

Constructors

Lookup

```
public Lookup(Connection connect) throws JdbcException
```

Class ecardfile.dbappl.LookupCache

```
Object
   |
   +---Cache
          |
          +---TableCache
                 |
                 +---ecardfile.dbappl.LookupCache
```

public class **LookupCache**
extends TableCache
Cache of the Lookup table
See Also:
TableCache

Variable Index

- ADDRESS
- EMAIL
- PHONE
- USERINFO

Constructor Index

- ecardfile.dbappl.LookupCache()
  Shouldn't use this constructor
- ecardfile.dbappl.LookupCache(JdbcConnectionBroker2, long)

Method Index

- addLookupTokens(Short, Hashtable)
  Put the Lookup tokens in the token Hashtable
- addVcardTokens(Short, Hashtable)
  Put the Vcard tokens in the token Hashtable
- getJdbcObject(Connection)
  This method should be defined in the sub class.
- populate()
  The database table is populated by using a JDBC object created by getJdbcObject.
- replaceLookupTokens(Short, String, int, Hashtable)
  Replace the Lookup tokens in the token Hashtable
- replaceVcardTokens(Short, String, int, Hashtable)
  Replace the Vcard tokens in the token Hashtable

Variables

- ADDRESS

```
public static final java.lang.Short ADDRESS
```

- EMAIL

```
public static final java.lang.Short EMAIL
```

- PHONE

```
public static final java.lang.Short PHONE
```

- USERINFO

```
public static final java.lang.Short USERINFO
```

Constructors

- LookupCache

```
public LookupCache()
```

Shouldn't use this constructor

- LookupCache

```
public LookupCache(JdbcConnectionBroker2 connMgr,
                   long secs) throws IOException
```

Parameters:

connMgr - - The JDBC Connection manager

secs - - The cache is repopulated every *secs* seconds

Methods

- addLookupTokens

```
public void addLookupTokens(Short category,
                            Hashtable tokens)
```

Put the Lookup tokens in the token Hashtable

- addVcardTokens

```
public void addVcardTokens(Short category,
                           Hashtable tokens)
```

Put the Vcard tokens in the token Hashtable

- getJdbcObject

```
protected multiserv.dbmgr.JdbcObject getJdbcObject(Connection
connect) throws JdbcException
```

This method should be defined in the sub class.

Overrides:

getJdbcObject in class TableCache

- populate

```
public void populate()
```

The database table is populated by using a JDBC object created by getJdbcObject. We call populate from the super class then keep the data in an optimized form for local use. The form is as for a token table of the types.

Overrides:

populate in class TableCache

- **replaceLookupTokens**

```
public void replaceLookupTokens(Short category,
                                String prefix,
                                int allocatedRows,
                                Hashtable tokens)
```

Replace the Lookup tokens in the token Hashtable

- **replaceVcardTokens**

```
public void replaceVcardTokens(Short category,
                               String prefix,
                               int allocatedRows,
                               Hashtable tokens)
```

Replace the Vcard tokens in the token Hashtable

Class ecardfile.dbappl.PersonalList

```
Object
   |
   +---JdbcObject
          |
          +---UserObject
                 |
                 +---ecardfile.dbappl.PersonalList
```

public class **PersonalList**
extends UserObject

Constructor Index

- ecardfile.dbappl.PersonalList(Connection)

Method Index

- Delete(DatabaseConnection2, long, long)
- DeleteByCardId(DatabaseConnection2, long)
  Note: Transaction should be setup around this method
- DeleteByUserId(DatabaseConnection2, long)
  Note: Transaction should be setup around this method
- Insert(DatabaseConnection2, long, long, long, short)
- IsCardThere(long, long)
- QueryContainsCard(long)
- QueryJoinByUserId(long)
- QueryJoinByUserIdName(long, char)

Constructors

PersonalList

```
public PersonalList(Connection connect) throws JdbcException
```

Methods

Delete

```
public int Delete(DatabaseConnection2 jdbc,
                  long personalListId,
                  long privateListId) throws JdbcException
```

DeleteByCardId

```
public int DeleteByCardId(DatabaseConnection2 jdbc,
                          long cardId) throws JdbcException
```

Note: Transaction should be setup around this method

- **DeleteByUserId**

```
public int DeleteByUserId(DatabaseConnection2 jdbc,
                          long userId) throws JdbcException
```

Note: Transaction should be setup around this method

- **Insert**

```
public long Insert(DatabaseConnection2 jdbc,
                   long listId,
                   long userId,
                   long cardId,
                   short mask) throws JdbcException
```

- **IsCardThere**

```
public boolean IsCardThere(long userId,
                           long cardId) throws JdbcException
```

- **QueryContainsCard**

```
public java.util.Vector QueryContainsCard(long cardId) throws
JdbcException
```

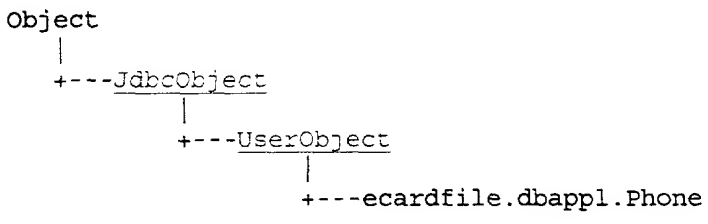
- **QueryJoinByUserId**

```
public java.util.Vector QueryJoinByUserId(long userId) throws
JdbcException
```

- **QueryJoinByUserIdName**

```
public java.util.Vector QueryJoinByUserIdName(long userId,
                                              char lastInitial)
throws JdbcException
```

Class ecardfile.dbappl.Phone



public class **Phone**
extends UserObject

Constructor Index

ecardfile.dbappl.Phone(Connection)

Constructors

Phone

```
public Phone(Connection connect) throws JdbcException
```

Class ecardfile.dbappl.PrivateList

```
Object
   |
   +---JdbcObject
          |
          +---UserObject
                 |
                 +---ecardfile.dbappl.PrivateList
```

public class **PrivateList**
extends UserObject

Constructor Index

- ecardfile.dbappl.PrivateList(Connection)

Method Index

- DeleteByCardId(long)
- Get(long, long)
- UpdateMask(long, short)

Constructors

- **PrivateList**

```
public PrivateList(Connection connect) throws JdbcException
```

Methods

- **DeleteByCardId**

```
public long DeleteByCardId(long cardId) throws JdbcException
```

- **Get**

```
public java.util.Hashtable Get(long loginId,
                               long cardId) throws JdbcException
```

- **UpdateMask**

```
public long UpdateMask(long privateListId,
                       short mask) throws JdbcException
```

Class ecardfile.dbappl.User

```
Object
   |
   +---JdbcObject
          |
          +---ecardfile.dbappl.User
```

public class **User**
extends JdbcObject

Constructor Index

- ecardfile.dbappl.User(Connection)

Method Index

- ConfirmUser(String, String)
- Delete(DatabaseConnection2, long)
- Get(String)
- GetForLogin(String, String)
- Insert(DatabaseConnection2, String[], Vector, Vector, Vector)
- QueryByFirstLastName(String, String)
- QueryByFirstLastNameSoundEx(String, String)
- Update(DatabaseConnection2, String[], Vector, Vector, Vector)

Constructors

- **User**

```
public User(Connection connect) throws JdbcException
```

Methods

- **ConfirmUser**

```
public int ConfirmUser(String eCardId,
                       String createId) throws JdbcException
```

- **Delete**

```
public int Delete(DatabaseConnection2 database,
                  long userId) throws JdbcException
```

- **Get**

```
public java.util.Hashtable Get(String szECardId) throws
JdbcException
```

- **GetForLogin**

```
public java.util.Hashtable GetForLogin(String szECardId,
                                       String szPassword) throws
JdbcException
```

- **Insert**

```
public long Insert(DatabaseConnection2 database,
                   String[] userRow,
                   Vector addressRows,
                   Vector phoneRows,
                   Vector emailRows) throws JdbcException
```

- **QueryByFirstLastName**

```
public java.util.Vector QueryByFirstLastName(String szFirstName,
                                             String szLastName)
throws JdbcException
```

- **QueryByFirstLastNameSoundEx**

```
public java.util.Vector QueryByFirstLastNameSoundEx(String
szFirstName,
                                                    String
szLastName) throws JdbcException
```

- **Update**

```
public long Update(DatabaseConnection2 database,
                   String[] userRow,
                   Vector addressRows,
                   Vector phoneRows,
                   Vector emailRows) throws JdbcException
```

Class ecardfile.dbappl.UserInfo

```
Object
   |
   +---JdbcObject
          |
          +---UserObject
                 |
                 +---ecardfile.dbappl.UserInfo
```

public class **UserInfo**
extends UserObject

Constructor Index

- ecardfile.dbappl.UserInfo(Connection)

Method Index

- QueryByCategory(long, short)
- Update(long, Vector)

Constructors

- **UserInfo**

```
public UserInfo(Connection connect) throws JdbcException
```

Methods

- **QueryByCategory**

```
public java.util.Vector QueryByCategory(long cardId,
                                        short category) throws
JdbcException
```

- **Update**

```
public long Update(long userId,
                   Vector rows) throws JdbcException
```

Class ecardfile.dbappl.UserObject

```
Object
   |
   +---JdbcObject
          |
          +---ecardfile.dbappl.UserObject
```

public class **UserObject**
extends JdbcObject

Constructor Index

- ecardfile.dbappl.UserObject(Connection, String, String[], int[], int[])

Method Index

- DeleteByUserId(long)
- Insert(long, String[])
- QueryByUserId(int)
- QueryByUserId(long)
- Update(long, String[])

Constructors

- **UserObject**

```
public UserObject(Connection connect,
                  String table,
                  String[] columnNames,
                  int columnLength,
                  int columnTypes) throws JdbcException
```

Methods

- **DeleteByUserId**

```
public long DeleteByUserId(long userId) throws JdbcException
```

- **Insert**

```
public long Insert(long userId,
                   String[] row) throws JdbcException
```

- **QueryByUserId**

```
public java.util.Vector QueryByUserId(int userId) throws
JdbcException
```

- **QueryByUserId**

```
public java.util.Vector QueryByUserId(long userId) throws
JdbcException
```

● Update

```
public long Update(long userId,
                   String[] row) throws JdbcException
```

Overrides:

Update in class JdbcObject

Class ecardfile.dbappl.WhereAmI

```
Object
   |
   +---JdbcObject
          |
          +---UserObject
                 |
                 +---ecardfile.dbappl.WhereAmI
```

public class **WhereAmI**
extends UserObject

Constructor Index

- ecardfile.dbappl.WhereAmI(Connection)

Method Index

- GetWithDate(long)
- QueryByExpiry(long, String)
- Update(long, String[])

Constructors

- **WhereAmI**

```
public WhereAmI(Connection connect) throws JdbcException
```

Methods

- **GetWithDate**

```
public java.util.Hashtable GetWithDate(long cardId) throws
JdbcException
```

- **QueryByExpiry**

```
public java.util.Vector QueryByExpiry(long cardId,
                                      String expDate) throws
JdbcException
```

- **Update**

```
public long Update(long userId,
                   String[] row) throws JdbcException
```

Overrides:
Update in class UserObject

package multiserv.applservlet

Class Index

- ApplServlet
- ApplicationInterface
- RequestHandler
- SessionTable

Class multiserv.applservlet.ApplServlet

```
Object
   |
   +---GenericServlet
           |
           +---HttpServlet
                   |
                   +---multiserv.applservlet.ApplServlet
```

public class **ApplServlet**
extends HttpServlet
The base class for all Servlets which use the Multi Server/Multi Servlet environment. Application servlets should extend this class.

For most cases, only the getApplicationInterface() needs to be overridden so that it returns the appropriate application specific implementation of ApplicationInterface.

Constructor Index

- multiserv.applservlet.ApplServlet()

Method Index

- decrCurrentCount()
- destroy()
- doGet(HttpServletRequest, HttpServletResponse)
  Overrides the doGet method provided by the HttpServlet superclass.
- doPost(HttpServletRequest, HttpServletResponse)
  Overrides the doPost method provided by the HttpServlet superclass.
- getApplicationInterface()
  Defines a method which returns an instance of ApplicationInterface.
- getProperty(String)
- getSessionMgr()
- incrCurrentCount()
- init(ServletConfig)
  Called when the servlet first gets loaded.
- log(String)
- setSessionMgrState(int)
- trace(String)

Constructors

- ApplServlet

```
public ApplServlet()
```

Methods

- **decrCurrentCount**

```
public synchronized void decrCurrentCount()
```

- **destroy**

```
public void destroy()
```

Overrides:
destroy in class GenericServlet

- **doGet**

```
public void doGet(HttpServletRequest req,
                  HttpServletResponse resp) throws
ServletException, IOException
```

Overrides the doGet method provided by the HttpServlet superclass. The service() method of HtpServlet handles the setup and dispatching to all doXXX() methods, which is why it usually should not be overridden.
Parameters:
req - The HTTP server request data.
resp - Provides methods to respond to the request.
Throws: ServletException
a problem occurred during the processing of the request.
Throws: IOException
an I/O problem was encountered.
Overrides:
doGet in class HttpServlet
See Also:
doPost

- **doPost**

```
public void doPost(HttpServletRequest req,
                   HttpServletResponse resp) throws
ServletException, IOException
```

Overrides the doPost method provided by the HttpServlet superclass. The service() method of HtpServlet handles the setup and dispatching to all doXXX() methods, which is why it usually should not be overridden.
Parameters:
req - The HTTP server request data.
resp - Provides methods to respond to the request.
Throws: ServletException
a problem occurred during the processing of the request.
Throws: IOException
an I/O problem was encountered.
Overrides:

doPost in class HttpServlet

See Also:

doGet

- **getApplicationInterface**

```
public multiserv.applservlet.ApplicationInterface
getApplicationInterface()
```

Defines a method which returns an instance of ApplicationInterface. This must be overridden in a subclass to provide an instance of a class which implements the methods defined in ApplicationInterface.

- **getProperty**

```
public java.lang.String getProperty(String propName)
```

- **getSessionMgr**

```
public synchronized multiserv.sessionmgr.SessionMgr
getSessionMgr() throws NotBoundException, RemoteException,
MalformedURLException
```

- **incrCurrentCount**

```
public synchronized void incrCurrentCount()
```

- **init**

```
public void init(ServletConfig config) throws ServletException
```

Called when the servlet first gets loaded. Do servlet initialization here. This specializes the init() method in GenericServlet.

Parameters:

config - servlet configuration information.

Throws: ServletException

a problem occurred during the initialization of the servlet.

Overrides:

init in class GenericServlet

- **log**

```
public void log(String msg)
```

Overrides:

log in class GenericServlet

- **setSessionMgrState**

```
public synchronized void setSessionMgrState(int newState)
```

- **trace**

```
public void trace(String msg)
```

Class multiserv.applservlet.ApplicationInterface

```
Object
   |
   +---RemoteObject
           |
           +---RemoteServer
                   |
                   +---UnicastRemoteObject
                           |
                           +---
multiserv.applservlet.ApplicationInterface
```

public abstract class **ApplicationInterface**
extends UnicastRemoteObject
implements SessionObserver
Encapsulates the application dependent operations which are invoked from the RequestHandler class.

Constructor Index

- multiserv.applservlet.ApplicationInterface()

Method Index

- accessDenied(String, HttpServletRequest, HttpServletResponse)
  Called when a received request does not contain a valid session id.
- authenticateUser(String, String, HttpServletRequest)
  Authenticate the user using a user id and password combination.
- chainRequest(String, HttpServletRequest, HttpServletResponse)
  Method to forward a URL to another servlet.
- checkAccess(HttpServletRequest)
  Check the HTTP request data and decide if access should be allowed.
- db_error(HttpServletRequest, HttpServletResponse)
  A database error has occurred
- db_error(HttpServletRequest, HttpServletResponse, String)
  A database error has occurred
- destroy()
  Hook to get the ApplicationInterface to brush its teeth before going to bed.
- doc_access_error(HttpServletRequest, HttpServletResponse)
  A document access error has occurred
- executeOperation(String, String, Session, HttpServletRequest, HttpServletResponse)
  Called by RequestHandler to execute an application defined operation.

- getOperation(HttpServletRequest)
  Determine the type of operation to be invoked by the request.
- getPassword(HttpServletRequest)
  Get a string representing the user's password
- getProperty(String)
- getServletParameter(HttpServletRequest, String)
  Convenience method to extract a parameter from a HttpServletRequest.
- getServletParameterValues(HttpServletRequest, String)
  Convenience method to a multi value parameter from a HttpServletRequest.
- getSessionId(HttpServletRequest)
  Extract and return the session id from the original HTTP request.
- getTimeZone()
- getUserId(HttpServletRequest)
  Get a string representing the user identification
- init(ApplServlet, String, String)
  Used to initialize the application specific class which implements this interface.
- io_error(HttpServletRequest, HttpServletResponse)
  An io exception has occurred
- log(String)
- nfe_error(HttpServletRequest, HttpServletResponse)
  A NumberFormatException has occurred
- nfe_error(HttpServletRequest, HttpServletResponse, String)
  A NumberFormatException has occurred
- notify(SessionNotification)
  Override this method to be receive notifications from the Session Manager.
- nse_error(HttpServletRequest, HttpServletResponse)
  A non serializable exception has occurred
- postAuthenticate(String, Session, HttpServletRequest, HttpServletResponse)
  Post authentication functionality.
- postDestroy(HttpServletRequest, HttpServletResponse)
  Called by the RequestHandler immediately after the destruction of the session object.
- preDestroy(String, Session, HttpServletRequest, HttpServletResponse)
  Called by RequestHandler prior to the destruction of the session object.
- re_error(HttpServletRequest, HttpServletResponse)
  A Remote Exception has occurred
- sae_error(HttpServletRequest, HttpServletResponse)
  A Session Access Exception has occurred
- sendError(HttpServletRequest, String, String, HttpServletResponse)
  General error handling routine.
- sessionFailure(String, HttpServletRequest)

Notify the application that an attempt to access a session using sessionId failed.

- trace(String)
- unknownOperation(HttpServletRequest, HttpServletResponse)

  An unknown operation has been requested
- validateSession(String, Session, HttpServletRequest)

  Validate the passed session This method can also be used to implement any security checks or other checks such as if undesirables are accessing the site - such as SPAMMERS

Constructors

- **ApplicationInterface**

```
public ApplicationInterface() throws RemoteException
```

Methods

- **accessDenied**

```
public abstract void accessDenied(String operation,
                                  HttpServletRequest req,
                                  HttpServletResponse resp)
```

Called when a received request does not contain a valid session id. The application should return an error/warning document to the user.

Parameters:

operation - the operation being performed
req - the original HTTP request.
resp - provides methods to respond to the HTTP server.
format - the http format to reply in

- **authenticateUser**

```
public abstract multiserv.sessionmgr.GenericSession
authenticateUser(String userId,
String password,
HttpServletRequest req) throws Exception
```

Authenticate the user using a user id and password combination. This method can also be used to implement any pre-session creation functionality.

Parameters:

userId - The user id string.
password - The user's password string.
req - The original HTTP request data.

Returns:

A session object containing any initial session data. The returned object must be a sub-class of GenericSession. This will be used to initialize the session object in the session manager. If the authentication fails, null is returned.

- **chainRequest**

```
public void chainRequest(String servletUrl,
                         HttpServletRequest req,
                         HttpServletResponse resp) throws
ServletException, IOException
```

Method to forward a URL to another servlet. This is kept as infrastructure so that it is easily changed to the most correct way of doing this.

Parameters:

servletUrl - The Servlet URL to which this request should be chained

req - The HTTP request from the current servlet which is being forwarded on.

resp - provides methods to respond to the HTTP server.

Throws: ServletException

Throws: IOException

See Also:

getServletParameter

- **checkAccess**

```
public abstract boolean checkAccess(HttpServletRequest req)
throws Exception
```

Check the HTTP request data and decide if access should be allowed. An application might want to check the IP address or other paramters in the request.

Parameters:

req - The original HTTP request data.

Returns:

An indication if access is to be granted

- **db_error**

```
public abstract void db_error(HttpServletRequest req,
                              HttpServletResponse resp)
```

A database error has occurred

Parameters:

format - the predefined format to reply in

resp - The HTTP response

- **db_error**

```
public abstract void db_error(HttpServletRequest req,
                              HttpServletResponse resp,
                              String msg)
```

A database error has occurred

Parameters:

format - the predefined format to reply in

resp - The HTTP response

- **destroy**

```
protected abstract void destroy()
```

Hook to get the ApplicationInterface to brush its teeth before going to bed.

- **doc_access_error**

```
public abstract void doc_access_error(HttpServletRequest req,
                                      HttpServletResponse resp)
```

A document access error has occurred

Parameters:

format - the predefined format to reply in

resp - The HTTP response

- **executeOperation**

```
public abstract void executeOperation(String operation,
                                      String sessionId,
                                      Session session,
                                      HttpServletRequest req,
                                      HttpServletResponse resp)
```

Called by RequestHandler to execute an application defined operation.

Parameters:

operation - Specifies the operation to perform. This should be one of the operation strings returned by getOperation.

session - An interface to the current session object.

sessionId - The session id string for the current session.

req - The data from the original HTTP request.

resp - Provides methods for responding to the request.

- **getOperation**

```
public abstract java.lang.String getOperation(HttpServletRequest
req)
```

Determine the type of operation to be invoked by the request. Note that session creation and session destruction requests are indicated by the operation strings defined by RequestHandler.CREATE and RequestHandler.DESTROY respectively.

Parameters:

req - The HTTP request.

Returns:

A string describing the operation to carry out.

- **getPassword**

```
public abstract java.lang.String getPassword(HttpServletRequest
req)
```

Get a string representing the user's password

Parameters:

req - The original HTTP request data.

Returns:

A string containing the password.

- **getProperty**

```
public java.lang.String getProperty(String propName)
```

- **getServletParameter**

```
public java.lang.String getServletParameter(HttpServletRequest
req,
                                            String parmName)
throws ServletException, IOException
```

Convenience method to extract a parameter from a HttpServletRequest. The method first checks to see if the parameter came from another servlet by trying to extract from the request using getAttribute(). If this fails it uses getParameter to extract the parameter. This order of checking means that a calling servlet can override parameters.

Parameters:
req - The HTTP request
parmName - The name of the parameter to extract
Returns:
The value of the parameter or null if not found
Throws: ServletException
Throws: IOException
See Also:
getServletParameter

- **getServletParameterValues**

```
public java.lang.String[]
getServletParameterValues(HttpServletRequest req,
                                                  String
parmName) throws ServletException, IOException
```

Convenience method to a multi value parameter from a HttpServletRequest. The method first checks to see if the parameter came from another servlet by trying to extract from the request using getAttribute(). If this fails it uses getParameterValues to extract the parameter. This order of checking means that a calling servlet can override parameters. The caller must be aware if the parameter is a multi value parameter.

Parameters:
req - The HTTP request
parmName - The name of the parameter to extract
Returns:
The value of the parameter or null if not found
Throws: ServletException
Throws: IOException
See Also:
getServletParameter

- **getSessionId**

```
public abstract java.lang.String getSessionId(HttpServletRequest
req) throws Exception
```

Extract and return the session id from the original HTTP request.

Parameters:

req - the original HTTP request.

Returns:

the session id associated with the request.

- **getTimeZone**

```
protected java.util.TimeZone getTimeZone()
```

- **getUserId**

```
public abstract java.lang.String getUserId(HttpServletRequest
req)
```

Get a string representing the user identification

Parameters:

req - The original HTTP request data.

Returns:

A string containing the user id.

- **init**

```
public void init(ApplServlet applServlet,
                 String managerName,
                 String rmiHost) throws ServletException,
IOException
```

Used to initialize the application specific class which implements this interface. The servlet configuration is passed in so that the servlet environment is available to the application code.

Parameters:

applServlet - The Servlet instance which owns this ApplicationInterface instance.

managerName - The name of the session manager instance to map.

rmihost - The host on which the registry is running

- **io_error**

```
public abstract void io_error(HttpServletRequest req,
                              HttpServletResponse resp)
```

An io exception has occurred

Parameters:

format - the predefined format to reply in

resp - The HTTP response

- **log**

```
public void log(String message)
```

- **nfe_error**

```
public abstract void nfe_error(HttpServletRequest req,
                               HttpServletResponse resp)
```

A NumberFormatException has occurred
Parameters:
format - the predefined format to reply in
resp - The HTTP response

- **nfe_error**

```
public abstract void nfe_error(HttpServletRequest req,
                               HttpServletResponse resp,
                               String msg)
```

A NumberFormatException has occurred
Parameters:
format - the predefined format to reply in
resp - The HTTP response
msg - Additional info to be displayed

- **notify**

```
public void notify(SessionNotification nofn)
```

Override this method to be receive notifications from the Session Manager.
Parameters:
nofn - Interface which accesses notification information.

- **nse_error**

```
public abstract void nse_error(HttpServletRequest req,
                               HttpServletResponse resp)
```

A non serializable exception has occurred
Parameters:
format - the predefined format to reply in
resp - The HTTP response

- **postAuthenticate**

```
public abstract void postAuthenticate(String sessionId,
                                      Session initialSession,
                                      HttpServletRequest req,
                                      HttpServletResponse resp)
throws Exception
```

Post authentication functionality. This would probably include sending an HTML document back to the user.
Parameters:
sessionId - a string identifying the new client session.
initialSession - the new client Session object initialized with initial data.
req - the original HTTP request.
resp - provides methods to respond to the HTTP server.

- **postDestroy**

```
public abstract void postDestroy(HttpServletRequest req,
                                 HttpServletResponse resp)
```

Called by the RequestHandler immediately after the destruction of the session object.

Parameters:

req - The data from the original HTTP request.

resp - Provides methods for responding to the request.

- **preDestroy**

```
public abstract void preDestroy(String sessionId,
                                Session session,
                                HttpServletRequest req,
                                HttpServletResponse resp)
```

Called by RequestHandler prior to the destruction of the session object.

Parameters:

sessionId - The session id string for the current session.

session - An interface to the associated session object.

req - The data from the original HTTP request.

resp - Provides methods for responding to the request.

- **re_error**

```
public abstract void re_error(HttpServletRequest req,
                              HttpServletResponse resp)
```

A Remote Exception has occurred

Parameters:

format - the predefined format to reply in

resp - The HTTP response

- **sae_error**

```
public abstract void sae_error(HttpServletRequest req,
                               HttpServletResponse resp)
```

A Session Access Exception has occurred

Parameters:

format - the predefined format to reply in

resp - The HTTP response

- **sendError**

```
public abstract void sendError(HttpServletRequest req,
                               String msg1,
                               String msg2,
                               HttpServletResponse resp)
```

General error handling routine. This will force the class user to implement error messages

Parameters:

req - The HTTP request

msg1 - First message to be displayed

msg2 - Second message to be displayed

resp - The HTTP response

- **sessionFailure**

```
public abstract boolean sessionFailure(String sessionId,
                                       HttpServletRequest req)
throws Exception
```

Notify the application that an attempt to access a sesssion using sessionId failed. This could be due to a Bogus sessionId or an expired session. The application will decide if a new session is to be created or if accessDenied() should be called. An application might want to log the failure or implement some sort of safeguard such as blocking the IP Address after a number of failures.

Parameters:

req - The original HTTP request data.

sessionId - The session id string for the current session.

Returns:

If true a new session will be created

- **trace**

```
public void trace(String message)
```

- **unknownOperation**

```
public abstract void unknownOperation(HttpServletRequest req,
                                      HttpServletResponse resp)
```

An unknown operation has been requested

Parameters:

format - the predefined format to reply in

resp - The HTTP response

- **validateSession**

```
public abstract boolean validateSession(String sessionId,
                                        Session session,
                                        HttpServletRequest req)
throws Exception
```

Validate the passed session This method can also be used to implement any security checks or other checks such as if undesirables are accessing the site - such as SPAMMERS

Parameters:

session - The session to validate.

req - The original HTTP request data.

Returns:

An indication if the session is valid

Class multiserv.applservlet.RequestHandler

```
Object
   |
   +---multiserv.applservlet.RequestHandler
```

public class **RequestHandler**
extends Object
A RequestHandler instance is created for each HTTP request received by the Application Servlet.

Variable Index

- CREATE
  The predefined operation types which can be executed by RequestHandler.
- DESTROY
- applInterface
  An instance of an application specific class which implements the ApplicationInterface
- managerName
  The name of the remote Session Manager being used
- operation
  Shows which operation this thread is executing
- request
  Holds the data from the original HTTP request
- response
  Provides methods for responding to the request
- rmiHost
  The hostname of the RMI registry
- servlet
  The parent Servlet instance which created this RequestHandler

Constructor Index

- multiserv.applservlet.RequestHandler(String, String, ApplServlet, ApplicationInterface, HttpServletRequest, HttpServletResponse)
  The constructor.

Method Index

- handle()
  The method which implements the request handling functionality.
- sessionMgr()

- sessionObjName(String)
  Constructs the session object name given the session id.

Variables

- CREATE

```
public static final java.lang.String CREATE
```

  The predefined operation types which can be executed by RequestHandler. All other operation types are application dependent and must be implemented using ApplicationInterface (or extension thereof)

- DESTROY

```
public static final java.lang.String DESTROY
```

- **applInterface**

```
protected multiserv.applservlet.ApplicationInterface
applInterface
```

  An instance of an application specific class which implements the ApplicationInterface

- **managerName**

```
protected java.lang.String managerName
```

  The name of the remote Session Manager being used

- **operation**

```
protected java.lang.String operation
```

  Shows which operation this thread is executing

- **request**

```
protected javax.servlet.http.HttpServletRequest request
```

  Holds the data from the original HTTP request

- **response**

```
protected javax.servlet.http.HttpServletResponse response
```

  Provides methods for responding to the request

- **rmiHost**

```
protected java.lang.String rmiHost
```

  The hostname of the RMI registry

- **servlet**

```
protected multiserv.applservlet.ApplServlet servlet
```

  The parent Servlet instance which created this RequestHandler

Constructors

- RequestHandler

```
public RequestHandler(String mgrName,
```

```
                        String rmihost,
                        ApplServlet applServlet,
                        ApplicationInterface appl,
                        HttpServletRequest req,
                        HttpServletResponse resp) throws
RemoteException, MalformedURLException, NotBoundException
```

The constructor. Initializes the instances data members.

Parameters:

mgrName - The name of the session manager instance to map.
rmihost - The host on which the registry is running
applServlet - The parent ApplServlet instance.
appl - The application specifics
req - The request data from the HTTP server.
resp - Provides methods for responding to the request.

Throws: RemoteException
if registry could not be contacted.

Throws: NotBoundException
if SessionMgr is not currently bound.

Methods

- **handle**

```
public void handle()
```

The method which implements the request handling functionality. This method processes requests and invokes the appropriate application level methods in ApplicationInterface.

- **sessionMgr**

```
protected multiserv.sessionmgr.SessionMgr sessionMgr() throws
NotBoundException, RemoteException, MalformedURLException
```

- **sessionObjName**

```
protected java.lang.String sessionObjName(String sid)
```

Constructs the session object name given the session id.

Parameters:

sid - the session id string for the object

Returns:

a string containing the session object name or the empty string if sid is null.

Class multiserv.applservlet.SessionTable

```
Object
   |
   +---Dictionary
           |
           +---Hashtable
                   |
                   +---multiserv.applservlet.SessionTable
```

public class **SessionTable**
extends Hashtable
A map containing the remote session object interfaces. Each interface is keyed using it's associated session id string.

Constructor Index

- multiserv.applservlet.SessionTable()

Method Index

- getSession(String)
  Retrieve a Session interface using the associated session id string.
- putSession(Session, String)
  Insert a Session interface into the map using the associated session id as the key.

Constructors

- SessionTable

```
public SessionTable()
```

Methods

- **getSession**

```
public multiserv.sessionmgr.Session getSession(String sessionId)
```

Retrieve a Session interface using the associated session id string.
Parameters:
sessionId - The session id string.
Returns:
The Session interface keyed to the given session id or null if the key is invalid.

- **putSession**

```
public void putSession(Session sess,
```

```
String sessionId)
```

Insert a Session interface into the map using the associated session id as the key.

Parameters:

sess - The remote session object interface.

sessionId - The session id string.

package multiserv.dbmgr

Class Index

- JdbcConnection
- JdbcConnectionBroker2
- JdbcConnectionFactory
- JdbcObject
- JdbcVendor
- TableCache
- Timing

Exception Index

- JdbcException

Class multiserv.dbmgr.JdbcConnection

```
Object
   |
   +---multiserv.dbmgr.JdbcConnection
```

public abstract class **JdbcConnection**
extends Object

Variable Index

- debug

Constructor Index

- multiserv.dbmgr.JdbcConnection()
  Default constructor only used for JdbcConnectionFactory
- multiserv.dbmgr.JdbcConnection(String, String, String)

Method Index

- Connect(String, String, String)
- Initialize()
- clearWarnings()
- close()
- createStatement()
- getConnection()
- getInstance(String, String, String)
- getWarnings()
- isClosed()
- toString()

Variables

debug

```
public static boolean debug
```

Constructors

JdbcConnection

```
public JdbcConnection()
```
Default constructor only used for JdbcConnectionFactory

JdbcConnection

```
protected JdbcConnection(String URL,
                         String username,
                         String password) throws JdbcException
```

Methods

- **Connect**

```
public boolean Connect(String URL,
                       String username,
                       String password)
```

- **Initialize**

```
protected abstract void Initialize() throws JdbcException
```

- **clearWarnings**

```
public void clearWarnings() throws SQLException
```

- **close**

```
public void close() throws SQLException
```

- **createStatement**

```
public java.sql.Statement createStatement() throws SQLException
```

- **getConnection**

```
public java.sql.Connection getConnection()
```

- **getInstance**

```
public abstract multiserv.dbmgr.JdbcConnection
getInstance(String URL,String username,String password) throws
JdbcException
```

- **getWarnings**

```
public java.sql.SQLWarning getWarnings() throws SQLException
```

- **isClosed**

```
public boolean isClosed() throws SQLException
```

- **toString**

```
public java.lang.String toString()
```

Overrides:
toString in class Object

Class multiserv.dbmgr.JdbcConnectionBroker2

```
Object
   |
   +----multiserv.dbmgr.JdbcConnectionBroker2
```

public class **JdbcConnectionBroker2**
extends Object
implements Runnable
JdbcConnectionBroker2 A servlet-based broker for database connections. Creates and manages a pool of database connections.

Version:

1.0.7 9/19/98

Author:

Marc A. Mnich

Constructor Index

multiserv.dbmgr.JdbcConnectionBroker2(String, String, String, String, JdbcConnection, int, int, String, double)
Creates a new Connection Broker
dbDriver: JDBC driver.

Method Index

- destroy()
  Shuts down the housekeeping thread and closes all connections in the pool.
- freeConnection(JdbcConnection)
  Frees a connection.
- getAge(JdbcConnection)
  Returns the age of a connection -- the time since it was handed out to an application.
- getConnection()
  This class hands out the connections in round-robin order.
- idOfConnection(JdbcConnection)
  Returns the local JDBC ID for a connection.
- interrupt()
  A hook for future expansion.
- release()
  Release was method used in previous Connection Pool manager
- run()
  Housekeeping thread.

Constructors

- JdbcConnectionBroker2

```
public JdbcConnectionBroker2(String dbDriver,
                             String dbServer,
                             String dbLogin,
                             String dbPassword,
                             JdbcConnection connection,
                             int minConns,
                             int maxConns,
                             String logFileString,
                             double maxConnTime) throws
IOException
```

Creates a new Connection Broker
dbDriver: JDBC driver. e.g. 'oracle.jdbc.driver.OracleDriver'
dbServer: JDBC connect string. e.g.
'jdbc:oracle:thin:@203.92.21.109:1526:orcl'
dbLogin: Database login name. e.g. 'Scott'
dbPassword: Database password. e.g. 'Tiger'
minConns: Minimum number of connections to start with.
maxConns: Maximum number of connections in dynamic pool.
logFileString: Absolute path name for log file. e.g. 'c:\\temp\\mylog.log'
maxConnTime: Time in days between connection resets. (Reset does a basic cleanup)

Methods

- destroy

```
public void destroy()
```

Shuts down the housekeeping thread and closes all connections in the pool. Call this method from the destroy() method of the servlet.

- freeConnection

```
public java.lang.String freeConnection(JdbcConnection conn)
```

Frees a connection. Replaces connection back into the main pool for resuse.

- getAge

```
public long getAge(JdbcConnection conn)
```

Returns the age of a connection -- the time since it was handed out to an application.

- getConnection

```
public multiserv.dbmgr.JdbcConnection getConnection()
```

This class hands out the connections in round-robin order. This prevents a faulty connection from locking up an application entirely. A browser 'refresh' will get the next connection while the faulty connection is cleaned

up by the housekeeping thread. If the min number of threads are ever exhausted, new threads are added up the the max thread count. Finally, if all threads are in use, this method waits 2 seconds and tries again, up to ten times. After that, it returns a null.

- **idOfConnection**

```
public int idOfConnection(JdbcConnection conn)
```

Returns the local JDBC ID for a connection.

- **interrupt**

```
public void interrupt()
```

A hook for future expansion. Currently it is used to interrupt the housekeeping thread.

- **release**

```
public void release()
```

Release was method used in previous Connection Pool manager

- **run**

```
public void run()
```

Housekeeping thread. Runs in the background with low CPU overhead. Connections are checked for warnings and closure and are periodically restarted. This thread is a catchall for corrupted connections and prevents the buildup of open cursors. (Open cursors result when the application fails to close a Statement). This method acts as fault tolerance for bad connection/statement programming.

Class multiserv.dbmgr.JdbcConnectionFactory

```
Object
   |
   +---multiserv.dbmgr.JdbcConnectionFactory
```

public class JdbcConnectionFactory
extends Object

Constructor Index

-multiserv.dbmgr.JdbcConnectionFactory(JdbcConnection, String, String, String)

Method Index

•getConnection()

Constructors

-JdbcConnectionFactory

```
public JdbcConnectionFactory(JdbcConnection connection,
                             String URL,
                             String username,
                             String password)
```

Methods

●getConnection

```
public multiserv.dbmgr.JdbcConnection getConnection() throws
JdbcException
```

Class multiserv.dbmgr.JdbcObject

```
Object
   |
   +---multiserv.dbmgr.JdbcObject
```

public class **JdbcObject**
extends Object
The base for a JdbcObject. When specifying the column names the tableId, if used, must be specified as the first column as *table*Id. E.g. For a table called Users and there is an id column it must be specified as the first column as UsersId.

Variable Index

Constructor Index

Method Index

Variables

- COLUMNS

```
protected static final int COLUMNS
```
- COLUMNS_INSERT

```
protected static final int COLUMNS_INSERT
```
- DEBUG

```
public static boolean DEBUG
```
- INSERTS

```
protected static final int INSERTS
```
- UPDATES

```
protected static final int UPDATES
```
- VALUES

```
protected static final int VALUES
```
- connect

```
protected java.sql.Connection connect
```
- ps

```
protected java.sql.PreparedStatement ps
```
- psAll

```
protected java.sql.PreparedStatement psAll
```
- psById

```
protected java.sql.PreparedStatement psById
```
- psDelete

```
protected java.sql.PreparedStatement psDelete
```
- psInsert

```
protected java.sql.PreparedStatement psInsert
```
- psMaxId

```
protected java.sql.PreparedStatement psMaxId
```
- psUpdate

```
protected java.sql.PreparedStatement psUpdate
```
- psUpdateCheck

```
protected java.sql.PreparedStatement psUpdateCheck
```

Constructors

- JdbcObject

```
public JdbcObject(Connection connect,
                  String table,
                  String[] columnNames,
                  int columnLength,
                  int columnTypes) throws JdbcException
```

Methods

- Delete

```
public long Delete(long id) throws JdbcException
```

- Execute

```
public long Execute() throws JdbcException
```

- Get

```
public java.util.Hashtable Get() throws JdbcException
```

- Get

```
public java.util.Hashtable Get(long lId) throws JdbcException
```

- Insert

```
public long Insert(String[] row) throws JdbcException
```

- Query

```
public java.util.Vector Query() throws JdbcException
```

- QueryAll

```
public java.util.Vector QueryAll() throws JdbcException
```

- TableName

```
public java.lang.String TableName()
```

- Update

```
public long Update(long id,
                   String[] row) throws JdbcException
```

- UpdateCheck

```
public long UpdateCheck(long id,
                        long checkId,
                        String[] row) throws JdbcException
```

- UpdateRow

```
public long UpdateRow(String[] row) throws JdbcException
```

- getColumnNames

```
public java.lang.String[] getColumnNames()
```

- getColumnString

```
public java.lang.String getColumnString(int iType)
```

Generate a string containing the column names given a particular operation type. Valid operation types are:
COLUMNS
All of the columns
COLUMNS_INSERT
Columns for an insert. Note that this will not include the tableId column if JdbcVendor.SUPPORT_ID is turned on
UPDATES
Columns for an update. Note that this will not include the tableId column if JdbcVendor.SUPPORT_ID is turned on
VALUES
Columns for a VALUES clause of an INSERT statement. Note that this will not include the tableId column if JdbcVendor.SUPPORT_ID is turned on

Parameters:
iType - Indicate the operation type for which the column string is being generated

- **getColumnTypes**

```
public int[] getColumnTypes()
```

- **getRow**

```
public java.util.Hashtable getRow(ResultSet results,
                                  ResultSetMetaData meta,
                                  int cols)
```

Class multiserv.dbmgr.JdbcVendor

```
Object
   |
   +---multiserv.dbmgr.JdbcVendor
```

public class **JdbcVendor**
extends Object
All of the vendor specific stuff goes in here. Things like supporting an automatically incrementing row id column. Edit this file and recompile ... yuk!

Variable Index

Constructor Index

Method Index

Variables

● **DUP_INDEX**

```
public static int DUP_INDEX
```

● **DUP_VALUE**

```
public static int DUP_VALUE
```

INFORMIX When inserting a row and a unique column constraint is violated this is the error code returned in getErrorCode() from the SQLException

● **SUPPORT_ID**

```
public static boolean SUPPORT_ID
```

Some call it SERIAL (eg Informix), some call it IDENTITY others just don't have it. Its a column which automatically creates a serial id for a row. Postgres doesn't have it but come to thing of it there is another sort of Id for a row - the details escape me at the moment but that could be an option in the Postgres rather than the getNextId() method

Constructors

● **JdbcVendor**

public JdbcVendor()

Methods

- **duplicateIndex**

public static boolean duplicateIndex(SQLException e)

- **getId**

public static long getId(PreparedStatement pStmt) throws SQLException

- **knownError**

public static boolean knownError(SQLException e)

- **setLockModeWait**

public static boolean setLockModeWait(Connection connect, int seconds) throws JdbcException

---

Class multiserv.dbmgr.TableCache

```
Object
   |
   +---Cache
          |
          +---multiserv.dbmgr.TableCache
```

public abstract class **TableCache**
extends Cache
This class provides a base class for caching a JdbcObject. Subclasses simply need to override the getJdbcObject(Connection) method

Version:

$Id: TableCache.java,v 1.4 1999/12/03 19:42:22 peter Exp $

See Also:

getJdbcObject, Cache

Constructor Index

- multiserv.dbmgr.TableCache()
- multiserv.dbmgr.TableCache(JdbcConnectionBroker2)
- multiserv.dbmgr.TableCache(JdbcConnectionBroker2, long)

Method Index

- getJdbcObject(Connection)
  This method should be defined in the sub class.
- getRow(long)
  Retrieve a row from the cached table by specifying the row id
- populate()
  The database table is populated by using a JDBC object created by getJdbcObject.
- reinitialize()
- repopulate()
  Simply calls populate.
- setConnManager(JdbcConnectionBroker2)

Constructors

TableCache

```
public TableCache()
```

TableCache

```
public TableCache(JdbcConnectionBroker2 connMgr) throws
IOException
```

Parameters:

connMgr - - The JDBC Connection broker

TableCache

```
public TableCache(JdbcConnectionBroker2 connMgr,
                  long secs) throws IOException
```

Parameters:

connMgr - - The JDBC Connection broker

secs - - The cache is repopulated every secs seconds

Methods

● **getJdbcObject**

```
protected abstract multiserv.dbmgr.JdbcObject
getJdbcObject(Connection connect) throws JdbcException
```

This method should be defined in the sub class. It should just create an instance of a subclasses JdbcObject, passing it connect.

Parameters:
connect - Database connection.

● **getRow**

```
public java.util.Hashtable getRow(long id) throws CacheException
```

Retrieve a row from the cached table by specifying the row id

Parameters:
id - The id of the row to be retrieved.

Returns:
Hashtable as returned by a JdbcObject single row query.

Throws: CacheException
- row not found in Cache

● **populate**

```
public void populate()
```

The database table is populated by using a JDBC object created by getJdbcObject.

Overrides:
populate in class Cache

● **reinitialize**

```
public void reinitialize()
```

Overrides:
reinitialize in class Cache

● **repopulate**

```
public void repopulate()
```

Simply calls populate. Obviously want to have a long cache check time. Could put in some sort of check to see when the table was last updated but this would be some sort of Vendor specific hook.

Overrides:
repopulate in class Cache

● **setConnManager**

```
public void setConnManager(JdbcConnectionBroker2 connMgr)
```

Parameters:
connMgr - - The JDBC Connection broker

All Packages Class Hierarchy This Package Previous Next Index

Class multiserv.dbmgr.Timing

```
Object
   |
   +---multiserv.dbmgr.Timing
```

public class **Timing**
extends Object

Constructor Index

- multiserv.dbmgr.Timing()

Method Index

- LogTiming(String)

Constructors

- Timing

```
public Timing()
```

Methods

- LogTiming

```
public void LogTiming(String szMessage)
```

Class multiserv.dbmgr.JdbcException

```
Object
   |
   +---Throwable
           |
           +---Exception
                   |
                   +---SQLException
                           |
                           +---multiserv.dbmgr.JdbcException
```

public class JdbcException
extends SQLException

Constructor Index

- multiserv.dbmgr.JdbcException()
- multiserv.dbmgr.JdbcException(String)
- multiserv.dbmgr.JdbcException(String, String, int)

Method Index

- isDuplicateValue()

Constructors

JdbcException

```
public JdbcException()
```

JdbcException

```
public JdbcException(String s)
```

JdbcException

```
public JdbcException(String message,
                     String state,
                     int error)
```

Methods

isDuplicateValue

```
public boolean isDuplicateValue()
```

package multiserv.sessionmgr

Interface Index

- Session
- SessionMgr
- SessionNotification
- SessionObserver
- SessionTags

Class Index

- GenericSession
- NotificationData
- ObserverMap
- SessionExpirer
- SessionIdFactory
- SessionImpl
- SessionMap
- SessionMgrConnection
- SessionMgrImpl
- SessionMgrShutdown
- SocketHandler

Exception Index

- SessionAccessException

Interface multiserv.sessionmgr.Session

public abstract interface **Session**
extends Remote
The Session remote interface is used to access the SessionImpl objects within the session manager.

Method Index

Methods

- **expire**

```
public abstract void expire() throws RemoteException
```

Mark this object as expired.

- **getDouble**

```
public abstract double getDouble(String fieldName) throws
SessionAccessException, RemoteException
```

Get the value of a double precision floating point field within the associated SessionImpl instance.

Parameters:

fieldName - A string specifying the name of the field to set.

Returns:

The value of the field.

Throws: SessionAccessException

Field doesn't exist or is private.

Throws: RemoteException

Communications error.

- **getInt**

```
public abstract int getInt(String fieldName) throws
SessionAccessException, RemoteException
```

Get the value of an integer type field within the associated SessionImpl instance.

Parameters:

fieldName - A string specifying the name of the field to set.

Returns:

The value of the field.

Throws: SessionAccessException

Field doesn't exist or is private.

Throws: RemoteException

Communications error.

- **getLastAccessed**

```
public abstract long getLastAccessed() throws RemoteException
```

Get the time at which this instance was last accessed.

Returns:

the time in milliseconds since EPOCH when this object was last accessed.

- **getLong**

```
public abstract long getLong(String fieldName) throws
SessionAccessException, RemoteException
```

Get the value of a long integer field within the associated SessionImpl instance.

Parameters:

fieldName - A string specifying the name of the field to set.

Returns:

The value of the field.

Throws: SessionAccessException

Field doesn't exist or is private.
Throws: RemoteException
Communications error.

- **getObject**

```
public abstract java.lang.Object getObject(String fieldName)
throws SessionAccessException, RemoteException,
NotSerializableException
```

Get the value of a field within the associated SessionImpl instance. The field type must be an extension of Object and must also implement the Serializable interface.
Parameters:
fieldName - A string specifying the name of the field to set.
Returns:
The value of the field.
Throws: SessionAccessException
Field doesn't exist or is private.
Throws: RemoteException
Communications error.
Throws: NotSerializableException
value doesn't implement Serializable

- **hasExpired**

```
public abstract boolean hasExpired() throws RemoteException
```

Determine if this object has expired
Returns:
true if the object has expired, false otherwise.

- **setDouble**

```
public abstract void setDouble(String fieldName,
                               double value) throws
SessionAccessException, RemoteException
```

Set the value of a double precision floating point field within the associated SessionImpl instance.
Parameters:
fieldName - A string specifying the name of the field to set.
value - The value to set the field to.
Throws: SessionAccessException
Field doesn't exist or is private.
Throws: RemoteException
Communications error.

- **setInt**

```
public abstract void setInt(String fieldName,
                            int value) throws
SessionAccessException, RemoteException
```

Set the value of an integer type field within the associated SessionImpl instance.

Parameters:
fieldName - A string specifying the name of the field to set.
value - The value to set the field to.
Throws: SessionAccessException
Field doesn't exist or is private.
Throws: RemoteException
Communications error.

- **setLong**

```
public abstract void setLong(String fieldName,
                             long value) throws
SessionAccessException, RemoteException
```

Set the value of a long integer field within the associated SessionImpl instance.

Parameters:
fieldName - A string specifying the name of the field to set.
value - The value to set the field to.
Throws: SessionAccessException
Field doesn't exist or is private.
Throws: RemoteException
Communications error.

- **setObject**

```
public abstract void setObject(String fieldName,
                               Object value) throws
SessionAccessException, RemoteException,
NotSerializableException
```

Set the value of a field within the associated SessionImpl instance. The field type must be an extension of Object and must also implement the Serializable interface.

Parameters:
fieldName - A string specifying the name of the field to set.
value - The value to set the field to.
Throws: SessionAccessException
Field doesn't exist or is private.
Throws: RemoteException
Communications error.
Throws: NotSerializableException
value doesn't implement Serializable

- **touch**

```
public abstract void touch() throws RemoteException
```

Mark the time when this object was last accessed as now.

Interface multiserv.sessionmgr.SessionMgr

public abstract interface SessionMgr
extends Remote
The SessionMgr remote interface is used to control the session manager.

Method Index

Methods

● closeSession

```
public abstract void closeSession(String sessionId) throws
RemoteException, MalformedURLException, NotBoundException
```

Remove an existing SessionImpl object from the session manager.

Parameters:
sessionId - A string which identifies the session to be removed.

Throws: RemoteException
if some communication failure occurs.

Throws: NotBoundException
the session which is being closed does not have it's interface bound to the RMI registry.

● initSession

```
public abstract java.lang.String initSession(Session session)
throws RemoteException, MalformedURLException,
SessionAccessException, NotSerializableException
```

Create a new SessionImpl object within the session manager.

Parameters:

session - Contains the initial session data.

Returns:

A string used to identify the new session in subsequent accesses to the session manager.

Throws: RemoteException

if some communication failure occurs.

- **register**

```
public abstract void register(String id) throws RemoteException
```

A remote entity registers interest in events taking place within the session manager by calling this method. After registering the session manager can notify the entity via the Servlet's SessionObserver interface.

Parameters:

id - A unique string identifying the registering entity

Throws: RemoteException

if some communication failure occurs.

- **sessionIdList**

```
public abstract java.util.Vector sessionIdList() throws
RemoteException
```

Return an array containing all the current Session object identifiers.

Returns:

A Vector instance containing the Session object id's.

Throws: RemoteException

if some communication failure occurs.

- **shutdown**

```
public abstract void shutdown(String id) throws RemoteException
```

A remote entity informs the Session Manager that it is prepared to have the Manager shutdown operations. The Session Manager will shutdown operation as soon as it receives shutdown notifications from all registered entities.

Parameters:

id - A unique string identifying the entity

Throws: RemoteException

if some communication failure occurs.

- **unregister**

```
public abstract void unregister(String id) throws
RemoteException
```

A remote entity calls this method to stop further notifications being sent by the session manager.

Parameters:

id - A unique string identifying the entity.

Throws: RemoteException

if some communication failure occurs.

Interface

multiserv.sessionmgr.SessionNotification

public abstract interface **SessionNotification**
Provides the interface through which the Session Manager can transfer notification data. Any class which implements this interface must also implement interface java.io.Serializable.

Variable Index

- MANAGER_RELOAD
- MANAGER_RUNNING
- MANAGER_SHUTDOWN
- MANAGER_STOPPING
- SESSION_EXPIRATION

Method Index

- reason()
  Returns an integer code representing the reason for the notification.
- sessionId()
  Returns a string giving the session id for the Session object which caused the notification to be issued.
- sessionObj()
  Returns a reference to a copy of the Session object which this NotificationData instance relates to.

Variables

● MANAGER_RELOAD

```
public static final int MANAGER_RELOAD
```

● MANAGER_RUNNING

```
public static final int MANAGER_RUNNING
```

● MANAGER_SHUTDOWN

```
public static final int MANAGER_SHUTDOWN
```

● MANAGER_STOPPING

```
public static final int MANAGER_STOPPING
```

● SESSION_EXPIRATION

```
public static final int SESSION_EXPIRATION
```

Methods

- **reason**

```
public abstract int reason()
```

Returns an integer code representing the reason for the notification. In general, these integer codes will be application dependent.

- **sessionId**

```
public abstract java.lang.String sessionId()
```

Returns a string giving the session id for the Session object which caused the notification to be issued. This will return null if the notification is not connected with any Session object.

- **sessionObj**

```
public abstract multiserv.sessionmgr.Session sessionObj()
```

Returns a reference to a copy of the Session object which this NotificationData instance relates to. For example, if this is a session expiry notification then this method will return a reference to a copy of the Session object which was removed from the Session Manager.

Interface multiserv.sessionmgr.SessionObserver

public abstract interface **SessionObserver**
extends Remote
Provides the interface via which the Session Manager can notify remote entities.

Method Index

- notify(SessionNotification)
  Called by the Session Manager to notify the entities which implement this interface.

Methods

- **notify**

```
public abstract void notify(SessionNotification nofn) throws
RemoteException
```

Called by the Session Manager to notify the entities which implement this interface.

Parameters:
nofn - Interface which accesses data relating to the notification.

Throws: RemoteException
if some communication failure occurs.

Interface multiserv.sessionmgr.SessionTags

public abstract interface SessionTags
Defines the constants used for basic sessions

Variable Index

- STATUS_TOKEN
- TERMINATE_TAG
- TIMEOUT_TAG
- TIMEOUT_TOKEN
- TIME_TOKEN
- TYPE_TAG
- TYPE_TOKEN
- URL_TAG
- URL_TOKEN
- USERID_TAG

  The following constants define the HTML tags which are used to identify data fields.

- USERNAME_TAG

Variables

- APPL_OPERATION

```
public static final java.lang.String APPL_OPERATION
```

- BROWSER_TAG

```
public static final java.lang.String BROWSER_TAG
```

- BROWSER_TOKEN

```
public static final java.lang.String BROWSER_TOKEN
```

- COMMENT_TOKEN

```
public static final java.lang.String COMMENT_TOKEN
```

- COMPUTER_TAG

```
public static final java.lang.String COMPUTER_TAG
```

- COMPUTER_TOKEN

```
public static final java.lang.String COMPUTER_TOKEN
```

- CONNECTION_TAG

```
public static final java.lang.String CONNECTION_TAG
```

- CONNECTION_TOKEN

```
public static final java.lang.String CONNECTION_TOKEN
```

- COOKIE_TAG

```
public static final java.lang.String COOKIE_TAG
```

- DATE_TOKEN

```
public static final java.lang.String DATE_TOKEN
```

- DOCNAME_TAG

```
public static final java.lang.String DOCNAME_TAG
```

●EMAIL_TAG

```
public static final java.lang.String EMAIL_TAG
```

●END_TOKEN

```
public static final java.lang.String END_TOKEN
```

●GUC_TOKEN

```
public static final java.lang.String GUC_TOKEN
```

●LC_OPT_PREFIX

```
public static final java.lang.String LC_OPT_PREFIX
```

●LOGIN_TAG

```
public static final java.lang.String LOGIN_TAG
```

●LOGOUT_OP

```
public static final java.lang.String LOGOUT_OP
```

●MSG1_TOKEN

```
public static final java.lang.String MSG1_TOKEN
```

●MSG2_TOKEN

```
public static final java.lang.String MSG2_TOKEN
```

●NAME_TAG

```
public static final java.lang.String NAME_TAG
```

●NAME_TOKEN

```
public static final java.lang.String NAME_TOKEN
```

●OPT_PREFIX

```
public static final java.lang.String OPT_PREFIX
```

●OP_TAG

```
public static final java.lang.String OP_TAG
```

●OP_TOKEN

```
public static final java.lang.String OP_TOKEN
```

●PASSWORD_TAG

```
public static final java.lang.String PASSWORD_TAG
```

●SESSIONID_TAG

```
public static final java.lang.String SESSIONID_TAG
```

●SESSIONID_TOKEN

```
public static final java.lang.String SESSIONID_TOKEN
```

●SESS_ACTIVE_TAG

public static final java.lang.String SESS_ACTIVE_TAG

- SESS_ID_TAG

public static final java.lang.String SESS_ID_TAG

- SESS_IP_ADDR_TAG

public static final java.lang.String SESS_IP_ADDR_TAG

- SESS_START_TAG

public static final java.lang.String SESS_START_TAG

- SESS_USER_ID_TAG

public static final java.lang.String SESS_USER_ID_TAG

- SESS_USER_TAG

public static final java.lang.String SESS_USER_TAG

- SPEED_TAG

public static final java.lang.String SPEED_TAG

- SPEED_TOKEN

public static final java.lang.String SPEED_TOKEN

- START_TOKEN

public static final java.lang.String START_TOKEN

- STATUS_TOKEN

public static final java.lang.String STATUS_TOKEN

- TERMINATE_TAG

public static final java.lang.String TERMINATE_TAG

- TIMEOUT_TAG

public static final java.lang.String TIMEOUT_TAG

- TIMEOUT_TOKEN

public static final java.lang.String TIMEOUT_TOKEN

- TIME_TOKEN

public static final java.lang.String TIME_TOKEN

- TYPE_TAG

public static final java.lang.String TYPE_TAG

- TYPE_TOKEN

public static final java.lang.String TYPE_TOKEN

- URL_TAG

public static final java.lang.String URL_TAG

- URL_TOKEN

```
public static final java.lang.String URL_TOKEN
```

- USERID_TAG

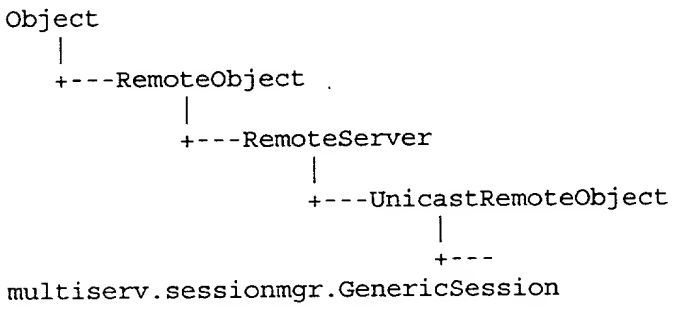
```
public static final java.lang.String USERID_TAG
```

The following constants define the HTML tags which are used to identify data fields. This file could possibly be generated automatically from or along with the HTML documents themselves.

- USERNAME_TAG

```
public static final java.lang.String USERNAME_TAG
```

Class multiserv.sessionmgr.GenericSession



public class **GenericSession**
extends UnicastRemoteObject
implements Session, Serializable, Cloneable

Variable Index

- expired
- lastAccessed

Constructor Index

- multiserv.sessionmgr.GenericSession()

Method Index

- expire()
  Mark this object as expired.
- getDouble(String)
  Get the value of a double precision floating point field within the associated SessionImpl instance.
- getInt(String)
  Get the value of an integer type field within the associated SessionImpl instance.
- getLastAccessed()
  Get the time at which this instance was last accessed.
- getLong(String)
  Get the value of a long integer field within the associated SessionImpl instance.
- getObject(String)
  Get the value of a field within the associated SessionImpl instance.
- hasExpired()
  Determine if this object has expired

- sessionFailure(Exception)
  General exception handler for the session object
- sessionFailure(String, Exception)
  General exception handler for the session object
- setDouble(String, double)
  Set the value of a double precision floating point field within the associated SessionImpl instance.
- setInt(String, int)
  Set the value of an integer type field within the associated SessionImpl instance.
- setLong(String, long)
  Set the value of a long integer field within the associated SessionImpl instance.
- setObject(String, Object)
  Set the value of a field within the associated SessionImpl instance.
- touch()
  Mark the time when this object was last accessed as now.

Variables

- expired

```
protected boolean expired
```

- lastAccessed

```
protected long lastAccessed
```

Constructors

- GenericSession

```
public GenericSession() throws RemoteException
```

Methods

- expire

```
public void expire() throws RemoteException
```

Mark this object as expired.

- getDouble

```
public double getDouble(String fieldName) throws
SessionAccessException
```

Get the value of a double precision floating point field within the associated SessionImpl instance.

Parameters:

fieldName - A string specifying the name of the field to set.

Returns:

The value of the field.

Throws: SessionAccessException

if there was a problem accessing the fields value.

- **getInt**

```
public int getInt(String fieldName) throws
SessionAccessException
```

Get the value of an integer type field within the associated SessionImpl instance.

Parameters:
fieldName - A string specifying the name of the field to set.
Returns:
The value of the field.
Throws: SessionAccessException
if there was a problem accessing the fields value.

- **getLastAccessed**

```
public long getLastAccessed() throws RemoteException
```

Get the time at which this instance was last accessed.

Returns:
the time in milliseconds since EPOCH when this object was last accessed.

- **getLong**

```
public long getLong(String fieldName) throws
SessionAccessException
```

Get the value of a long integer field within the associated SessionImpl instance.

Parameters:
fieldName - A string specifying the name of the field to set.
Returns:
The value of the field.
Throws: SessionAccessException
if there was a problem accessing the fields value.

- **getObject**

```
public java.lang.Object getObject(String fieldName) throws
SessionAccessException
```

Get the value of a field within the associated SessionImpl instance. The field type must be an extension of Object and must also implement the Serializable interface.

Parameters:
fieldName - A string specifying the name of the field to set.
Returns:
The value of the field.
Throws: SessionAccessException
if there was a problem accessing the fields value.

- **hasExpired**

```
public boolean hasExpired() throws RemoteException
```

Determine if this object has expired

Returns:

true if the object has expired, false otherwise.

- **sessionFailure**

```
protected void sessionFailure(Exception e) throws
SessionAccessException
```

General exception handler for the session object

Parameters:

e - the exception which is being handled

Throws: SessionAccessException

there was a problem while accessing a field.

- **sessionFailure**

```
protected void sessionFailure(String fieldName,
                              Exception e) throws
SessionAccessException
```

General exception handler for the session object

Parameters:

e - the exception which is being handled

Throws: SessionAccessException

there was a problem while accessing a field.

- **setDouble**

```
public void setDouble(String fieldName,
                      double value) throws
SessionAccessException
```

Set the value of a double precision floating point field within the associated SessionImpl instance.

Parameters:

fieldName - A string specifying the name of the field to set.

value - The value to set the field to.

Throws: SessionAccessException

if there was a problem accessing the fields value.

- **setInt**

```
public void setInt(String fieldName,
                   int value) throws SessionAccessException
```

Set the value of an integer type field within the associated SessionImpl instance.

Parameters:

fieldName - A string specifying the name of the field to set.

value - The value to set the field to.

Throws: SessionAccessException

if there was a problem accessing the fields value.

- **setLong**

```
public void setLong(String fieldName,
                    long value) throws SessionAccessException
```

Set the value of a long integer field within the associated SessionImpl instance.

Parameters:

fieldName - A string specifying the name of the field to set.

value - The value to set the field to.

Throws: SessionAccessException

if there was a problem accessing the fields value.

- **setObject**

```
public void setObject(String fieldName,
                      Object value) throws
SessionAccessException
```

Set the value of a field within the associated SessionImpl instance. The field type must be an extension of Object and must also implement the Serializable interface.

Parameters:

fieldName - A string specifying the name of the field to set.

value - The value to set the field to.

Throws: SessionAccessException

if there was a problem accessing the fields value.

- **touch**

```
public void touch() throws RemoteException
```

Mark the time when this object was last accessed as now.

Class multiserv.sessionmgr.NotificationData

```
Object
   |
   +---multiserv.sessionmgr.NotificationData
```

public class **NotificationData**
extends Object
implements SessionNotification, Serializable
Provides an implementation of the SessionNotification interface

Constructor Index

- multiserv.sessionmgr.NotificationData(int, String, Session)

Method Index

- reason()
  Returns an integer code representing the reason for the notification.
- sessionId()
  Returns a string giving the session id for the Session object which caused the notification to be issued.
- sessionObj()
  Returns a reference to a copy of the Session object which this NotificationData instance relates to.

Constructors

- **NotificationData**

```
public NotificationData(int code,
                        String id,
                        Session sess)
```

Methods

- **reason**

```
public int reason()
```

Returns an integer code representing the reason for the notification. In general, these integer codes will be application dependent.

- **sessionId**

```
public java.lang.String sessionId()
```

Returns a string giving the session id for the Session object which caused the notification to be issued. This will return null if the notification is not connected with any Session object.

● sessionObj

```
public multiserv.sessionmgr.Session sessionObj()
```

Returns a reference to a copy of the Session object which this NotificationData instance relates to. For example, if this is a session expiry notification then this method will return a reference to a copy of the Session object which was removed from the Session Manager.

Class multiserv.sessionmgr.ObserverMap

```
Object
   |
   +---Dictionary
           |
           +---Hashtable
                   |
                   +---multiserv.sessionmgr.ObserverMap
```

public class **ObserverMap**
extends Hashtable
The ObserverMap class stores the currently actice SessionObserver instances within the session manager.

Constructor Index

- multiserv.sessionmgr.ObserverMap()
- multiserv.sessionmgr.ObserverMap(int)
- multiserv.sessionmgr.ObserverMap(int, float)

Method Index

- putObserver(String)
  Insert an observer object within the map using the remote entity id string as the key.
- removeObserver(String)
  Remove an observer object from the map.

Constructors

ObserverMap

```
public ObserverMap()
```

ObserverMap

```
public ObserverMap(int capacity)
```

ObserverMap

```
public ObserverMap(int capacity,
                   float loadFactor)
```

Methods

putObserver

```
public void putObserver(String id)
```

Insert an observer object within the map using the remote entity id string as the key.

Parameters:

obs - The observer object to insert in the map. It must be implement the SessionObserver interface.

id - The remote entity id string.

- **removeObserver**

```
public void removeObserver(String id)
```

Remove an observer object from the map.

Parameters:

id - The id string for the observer to remove

Class multiserv.sessionmgr.SessionExpirer

```
Object
   |
   +---Thread
           |
           +---multiserv.sessionmgr.SessionExpirer
```

public class **SessionExpirer**
extends Thread
A thread object which expires stale session objects within the session manager at regular intervals.

Constructor Index

- multiserv.sessionmgr.SessionExpirer(SessionMgrImpl)

Method Index

- haltExpiries()
- run()

Constructors

- SessionExpirer

```
public SessionExpirer(SessionMgrImpl mgr)
```

Methods

- **haltExpiries**

```
public void haltExpiries()
```

- **run**

```
public void run()
```

Overrides:
run in class Thread

Class multiserv.sessionmgr.SessionIdFactory

```
Object
   |
   +---multiserv.sessionmgr.SessionIdFactory
```

public class **SessionIdFactory**
extends Object

Constructor Index

- multiserv.sessionmgr.SessionIdFactory()

Method Index

- createSessionId()
  Creates a randomly generated string with NUMDIG digits, NUMLOWER lowercase characters, and NUMUPPER characters arranged randomly.

Constructors

- **SessionIdFactory**

```
public SessionIdFactory()
```

Methods

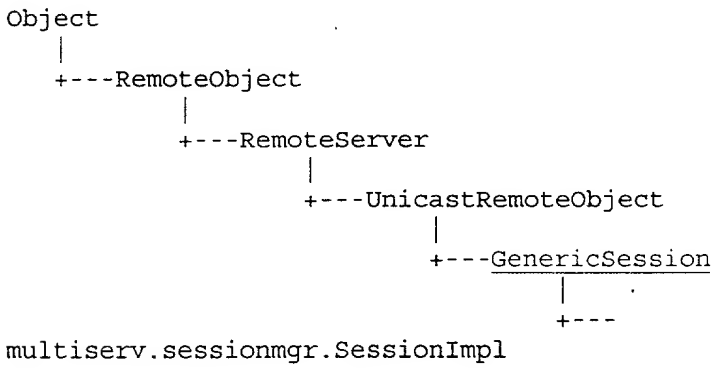
- **createSessionId**

```
public static java.lang.String createSessionId()
```

Creates a randomly generated string with NUMDIG digits, NUMLOWER lowercase characters, and NUMUPPER characters arranged randomly.

Returns:
the randomly generated string

Class multiserv.sessionmgr.SessionImpl



public class **SessionImpl**
extends GenericSession
implements SessionTags
Contains the application dependent data fields for the session.

Variable Index

- sess_active
- sess_ip_addr
- sess_start
- sess_user

  The actual Session Object members
- sess_user_id

Constructor Index

- multiserv.sessionmgr.SessionImpl()

  Get a new SessionImpl object within the session manager.
- multiserv.sessionmgr.SessionImpl(String, Long, BigDecimal, BigDecimal, String)

  Get a new SessionImpl object within the session manager.
- multiserv.sessionmgr.SessionImpl(Session)

  Get a new SessionImpl object within the session manager.

Method Index

- getSession()

  Get a new SessionImpl object within the session manager.
- getSession(String, Long, BigDecimal, BigDecimal, String)

  Get a new SessionImpl object within the session manager.
- getSession(Session)

Get a new SessionImpl object within the session manager.

Variables

- **sess_active**

```
protected java.math.BigDecimal sess_active
```

- **sess_ip_addr**

```
protected java.lang.String sess_ip_addr
```

- **sess_start**

```
protected java.math.BigDecimal sess_start
```

- **sess_user**

```
protected java.lang.String sess_user
```

The actual Session Object members

- **sess_user_id**

```
protected java.lang.Long sess_user_id
```

Constructors

- **SessionImpl**

```
public SessionImpl() throws RemoteException
```

Get a new SessionImpl object within the session manager. Gives the user the chance to override the default session behaviour.

Returns:

A string used to identify the new session in subsequent accesses to the session manager.

- **SessionImpl**

```
public SessionImpl(String sess_user,
                   Long sess_user_id,
                   BigDecimal sess_start,
                   BigDecimal sess_active,
                   String sess_ip_addr) throws RemoteException
```

Get a new SessionImpl object within the session manager. Gives the user the chance to override the default session behaviour.

Parameters:

sess_user - The user
sess_user_id - The user id
sess_start - Time session was started
sess_active - Last time the session was active
sess_ip_addr - Remote IP Address

Returns:

A string used to identify the new session in subsequent accesses to the session manager.

- **SessionImpl**

```
public SessionImpl(Session initial) throws RemoteException,
SessionAccessException, NotSerializableException
```

Get a new SessionImpl object within the session manager. Gives the user the chance to override the default session behaviour.

Parameters:

initial - The initial session

Returns:

A string used to identify the new session in subsequent accesses to the session manager.

Methods

- **getSession**

```
public static multiserv.sessionmgr.SessionImpl getSession()
throws RemoteException, SessionAccessException,
NotSerializableException
```

Get a new SessionImpl object within the session manager. Gives the user the chance to override the default session behaviour.

Returns:

A string used to identify the new session in subsequent accesses to the session manager.

- **getSession**

```
public static multiserv.sessionmgr.SessionImpl getSession(String
sess_user, Long sess_user_id,
BigDecimal sess_start,BigDecimal sess_active,
String sess_ip_addr) throws RemoteException,
SessionAccessException, NotSerializableException
```

Get a new SessionImpl object within the session manager. Gives the user the chance to override the default session behaviour.

Parameters:

sess_user - The user name
sess_user_id - The user id
sess_start - Time session was started
sess_active - Last time the session was active
sess_ip_addr - Remote IP Address

Returns:

A string used to identify the new session in subsequent accesses to the session manager.

- **getSession**

```
public static multiserv.sessionmgr.SessionImpl
getSession(Session initial) throws RemoteException,
SessionAccessException, NotSerializableException
```

Get a new SessionImpl object within the session manager. Gives the user the chance to override the default session behaviour.

Parameters:

session - Contains the initial session data.

Returns:

A string used to identify the new session in subsequent accesses to the session manager.

Class multiserv.sessionmgr.SessionMap

```
Object
   |
   +---Dictionary
           |
           +---Hashtable
                   |
                   +---multiserv.sessionmgr.SessionMap
```

public class **SessionMap**
extends Hashtable
The SessionMap class stores the currently active SessionImpl instances within the session manager.

Constructor Index

- multiserv.sessionmgr.SessionMap()
- multiserv.sessionmgr.SessionMap(int)
- multiserv.sessionmgr.SessionMap(int, float)

Method Index

- getSession(String)
  Retrieve a session object from within the using the specified session id key.
- putSession(Session, String)
  Insert a session object within the map using the session id string as the key.
- removeSession(String)
  Remove a session object from the map.

Constructors

SessionMap

```
public SessionMap()
```

SessionMap

```
public SessionMap(int capacity)
```

SessionMap

```
public SessionMap(int capacity,
                  float loadFactor)
```

Methods

getSession

```
public multiserv.sessionmgr.Session getSession(String sessionId)
```

Retrieve a session object from within the using the specified session id key.

Parameters:

sessionId - The session id string.

Returns:

The session object if the session id is valid otherwise null.

- **putSession**

```
public void putSession(Session session,
                       String sessionId)
```

Insert a session object within the map using the session id string as the key.

Parameters:

session - The session object to insert in the map. It must be a subclass of GenericSession.

sessionId - The session id string.

- **removeSession**

```
public void removeSession(String sessionId)
```

Remove a session object from the map.

Parameters:

sessionId - The id string for the session to remove

Class
multiserv.sessionmgr.SessionMgrConnection

```
Object
   |
   +---Thread
          |
          +---multiserv.sessionmgr.SessionMgrConnection
```

public class SessionMgrConnection
extends Thread

Constructor Index

- multiserv.sessionmgr.SessionMgrConnection(SessionMgrImpl, Socket)

Method Index

- run()

Constructors

- SessionMgrConnection

```
public SessionMgrConnection(SessionMgrImpl mgr,
                            Socket clientSocket)
```

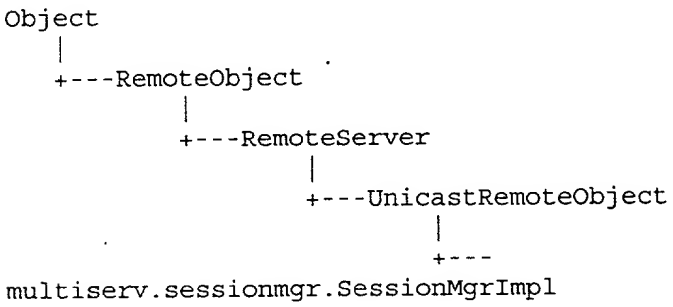
Methods

- run

```
public void run()
```

Overrides:
run in class Thread

Class multiserv.sessionmgr.SessionMgrImpl



public class **SessionMgrImpl**
extends UnicastRemoteObject
implements SessionMgr
The SessionMgr remote interface is used to control the session manager.

Variable Index

- config
  Contains the application configuration information

Constructor Index

- multiserv.sessionmgr.SessionMgrImpl(String, String, String)
  Construct a new instance of SessionMgrImpl.

Method Index

- closeSession(String)
  Remove an existing SessionImpl object from the session manager.
- expireOldSessions()
- finalize()
- initSession(Session)
  Create a new SessionImpl object within the session manager.
- log(String)
  Write information to stdout tagged with the date and time.
- main(String[])
  The main method for the session manager process.
- notifyServlets(int, String, Session)
- register(String)
  A remote entity registers interest in events taking place within the session manager by calling this method.
- reload()
- restoreState()

Restores the contents of the Session map from persistent store.

- saveState()

  Saves the contents of the Session map to persistent store.
- sessionIdList()

  Return an array containing all the current Session object identifiers.
- sessionObjName(String)

  Constructs the session object name given the session id.
- shutdown(String)

  A remote entity informs the Session Manager that it is prepared to have the Manager shutdown operations.
- shuttingdown()
- trace(String)
- unregister(String)

  A remote entity calls this method to stop further notifications being sent by the session manager.

Variables

- **config**

```
public multiserv.config.Config config
```

Contains the application configuration information

Constructors

- **SessionMgrImpl**

```
public SessionMgrImpl(String rmihost,
                      String name,
                      String configFile) throws RemoteException,
MalformedURLException
```

Construct a new instance of SessionMgrImpl.

Parameters:

managerName - Identifies this session manager instance

name - The name by which the session manager will be known

configFile - The configuration file

Methods

- **closeSession**

```
public void closeSession(String sessionId) throws
RemoteException, MalformedURLException, NotBoundException
```

Remove an existing SessionImpl object from the session manager.

Parameters:

sessionId - A string which identifies the session to be removed.

Throws: RemoteException

some communication problem occurred.

Throws: NotBoundException

the session which is being closed does not have it's interface bound to the RMI registry.

- **expireOldSessions**

```
public void expireOldSessions()
```

- **finalize**

```
protected void finalize()
```

Overrides:

finalize in class Object

- **initSession**

```
public java.lang.String initSession(Session initial) throws
RemoteExceptionMalformedURLException, SessionAccessException,
NotSerializableException
```

Create a new SessionImpl object within the session manager.

Parameters:

session - Contains the initial session data.

Returns:

A string used to identify the new session in subsequent accesses to the session manager.

- **log**

```
public void log(String message)
```

Write information to stdout tagged with the date and time.

Parameters:

message - the information to output

- **main**

```
public static void main(String[] args)
```

The main method for the session manager process. This installs a security manager as well as creating a registry so that the clients can lookup the manager objects. The session manager must be started with a single command line argument. This argument specifies the session manager name.

- **notifyServlets**

```
protected void notifyServlets(int reason,
                              String sid,
                              Session sess) throws
RemoteException
```

- **register**

```
public void register(String id) throws RemoteException
```

A remote entity registers interest in events taking place within the session manager by calling this method. After this the session manager can notify the entity via the Servlet's SessionObserver interface.

Parameters:

id - A unique string identifying the registering entity. The name should be an RMI addressable ID. That is, you should be able to tack "rmi://" on to the front of it. Suggestion is "host/servletId"
Throws: RemoteException
if some communication failure occurs.

- **reload**

```
public void reload() throws RemoteException
```

- **restoreState**

```
protected void restoreState()
```

Restores the contents of the Session map from persistent store.

- **saveState**

```
protected void saveState()
```

Saves the contents of the Session map to persistent store.

- **sessionIdList**

```
public java.util.Vector sessionIdList() throws RemoteException
```

Return an array containing all the current Session object identifiers.
Returns:
An Vector instance containing the Session object id's.
Throws: RemoteException
if some communication failure occurs.

- **sessionObjName**

```
protected java.lang.String sessionObjName(String sid)
```

Constructs the session object name given the session id.
Parameters:
sid - the session id string for the object
Returns:
a string containing the session object name or the empty string if sid is null.

- **shutdown**

```
public void shutdown(String id) throws RemoteException
```

A remote entity informs the Session Manager that it is prepared to have the Manager shutdown operations. The Session Manager will shutdown operation as soon as it receives shutdown notifications from all registered entities.
Parameters:
id - A unique string identifying the entity
Throws: RemoteException
if some communication failure occurs.

- **shuttingdown**

```
public void shuttingdown()
```

● **trace**

```
public void trace(String message)
```

● **unregister**

```
public void unregister(String id) throws RemoteException
```

A remote entity calls this method to stop further notifications being sent by the session manager.

Parameters:

id - A unique string identifying the entity.

Throws: RemoteException

if some communication failure occurs.

See Also:

register

Class multiserv.sessionmgr.SessionMgrShutdown

```
Object
   |
   +---Thread
           |
           +---multiserv.sessionmgr.SessionMgrShutdown
```

public class **SessionMgrShutdown**
extends Thread
A thread object which shuts down the SessionMgr process

Constructor Index

- multiserv.sessionmgr.SessionMgrShutdown()

Method Index

- run()

Constructors

- SessionMgrShutdown

```
public SessionMgrShutdown()
```

Methods

- **run**

```
public void run()
```

Overrides:
run in class Thread

Class multiserv.sessionmgr.SocketHandler

```
Object
   |
   +---Thread
           |
           +---multiserv.sessionmgr.SocketHandler
```

public class **SocketHandler**
extends Thread
A thread object which listens on a socket for connection attempts and then services commands from authorized clients. Normally only the host on which the session manager is running or the loopback device are allowed. The property multiserv.sessionmgr.allowedIPs of IPAddresses allows other hosts access - it should coonsist of white space separated IP addresses.

Constructor Index

- multiserv.sessionmgr.SocketHandler(SessionMgrImpl)
  Create a socket handler

Method Index

- halt()
- run()

Constructors

- **SocketHandler**

```
public SocketHandler(SessionMgrImpl mgr)
```

Create a socket handler
Parameters:
mgr - Reference to the session manager instance

Methods

- **halt**

```
public void halt()
```

- **run**

```
public void run()
```

Overrides:
run in class Thread

Class
multiserv.sessionmgr.SessionAccessException

```
Object
   |
   +---Throwable
           |
           +---Exception
                   |
                   +---
multiserv.sessionmgr.SessionAccessException
```

public class **SessionAccessException**
extends Exception
This exception is thrown to indicate a problem with accessing data within a session object.

Constructor Index

- multiserv.sessionmgr.SessionAccessException(String)
  Constructs a new SessionAccessException with the specified descriptive message.

Constructors

- **SessionAccessException**

```
public SessionAccessException(String msg)
```

Constructs a new SessionAccessException with the specified descriptive message.

package multiserv.util

Interface Index

- Watchable

Class Index

- Cache
- FileCache
- FileCacheFactory
- FileSuffixFilter
- HTMLCache
- HTMLCacheFactory
- HTMLDocument
- Logger
- SendMail
- SoundEx
- TimedCounter
- Watcher

Exception Index

- CacheException

Interface multiserv.util.Watchable

public abstract interface Watchable

Method Index

Methods

●wakeup

```
public abstract void wakeup()
```

●watch

```
public abstract void watch()
```

Class multiserv.util.Cache

```
Object
   |
   +---multiserv.util.Cache
```

public abstract class **Cache**
extends Object
implements Watchable

Variable Index

- DEBUG

Constructor Index

- multiserv.util.Cache()
- multiserv.util.Cache(long)

Method Index

- addObject(String, Object, long)
- getKeys()
  Get the keys of the table rows stored in the Cache.
- getObject(String)
- interruptWatching()
- populate()
- reinitialize()
- removeAll()
- removeObject(String)
- repopulate()
- setWatcherDelay(long)
- startWatching()
- stopWatching()
- updateObject(String, Object, long)
- wakeup()
- watch()

Variables

● DEBUG

```
public static boolean DEBUG
```

Constructors

- Cache

```
public Cache()
```
- Cache

```
public Cache(long secs)
```

Methods

- addObject

```
public synchronized void addObject(String name,
                                   Object obj,
                                   long lastModified)
```

- getKeys

```
protected synchronized java.util.Enumeration getKeys()
```

Get the keys of the table rows stored in the Cache.

Returns:

An Enumeration of the available keys. The keys are stored as strings.

- getObject

```
public synchronized java.lang.Object getObject(String name)
throws CacheException
```

- interruptWatching

```
public void interruptWatching()
```

- populate

```
public abstract void populate()
```

- reinitialize

```
public abstract void reinitialize()
```

- removeAll

```
public synchronized void removeAll()
```

- removeObject

```
public synchronized void removeObject(String name)
```

- repopulate

```
public abstract void repopulate()
```

- setWatcherDelay

```
public void setWatcherDelay(long secs)
```

- startWatching

```
public void startWatching()
```

- stopWatching

```
public void stopWatching()
```

- updateObject

```
public synchronized void updateObject(String name,
                                       Object obj,
                                       long lastModified)
```

- **wakeup**

```
public final synchronized void wakeup()
```

- **watch**

```
public final synchronized void watch()
```

Class multiserv.util.FileCache

```
Object
   |
   +---Cache
          |
          +---multiserv.util.FileCache
```

public class **FileCache**
extends Cache

Method Index

- getDocument(String)
- main(String[])
- populate()
- reinitialize()
- repopulate()
- setSuffixes(String[])

Methods

getDocument

```
public java.lang.String getDocument(String name) throws
CacheException
```

main

```
public static void main(String[] args)
```

populate

```
public void populate()
```

Overrides:
populate in class Cache

reinitialize

```
public void reinitialize()
```

Overrides:
reinitialize in class Cache

repopulate

```
public void repopulate()
```

Overrides:
repopulate in class Cache

setSuffixes

```
public void setSuffixes(String[] fileSuffixes)
```

Class multiserv.util.FileCacheFactory

```
Object
   |
   +---multiserv.util.FileCacheFactory
```

public class **FileCacheFactory**
extends Object

Constructor Index

- multiserv.util.FileCacheFactory()

Method Index

- getCache(String)
- getCache(String, String, long, String[])

Constructors

- FileCacheFactory

```
public FileCacheFactory()
```

Methods

- getCache

```
public static multiserv.util.FileCache getCache(String
cacheName) throws IOException
```

- getCache

```
public static multiserv.util.FileCache getCache(String
cacheName,
                                                String dir,
                                                long
cachePeriod,
                                                String[]
suffixes) throws IOException
```

Class multiserv.util.FileSuffixFilter

```
Object
   |
   +---multiserv.util.FileSuffixFilter
```

public class **FileSuffixFilter**
extends Object
implements FilenameFilter

Constructor Index

- multiserv.util.FileSuffixFilter(String[])
- multiserv.util.FileSuffixFilter(String[], long)

Method Index

- accept(File, String)

Constructors

- **FileSuffixFilter**

```
public FileSuffixFilter(String[] suffixes)
```

- **FileSuffixFilter**

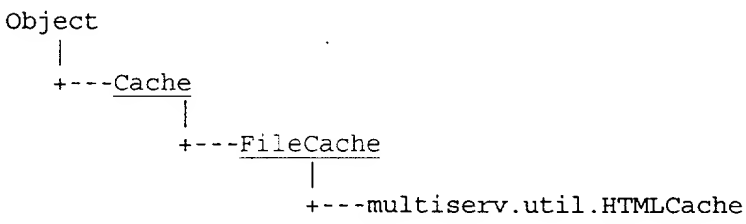
```
public FileSuffixFilter(String[] suffixes,
                        long modifiedSince)
```

Methods

- **accept**

```
public boolean accept(File dir,
                      String name)
```

Class multiserv.util.HTMLCache



public class **HTMLCache**
extends FileCache

Class multiserv.util.HTMLCacheFactory

```
Object
   |
   +---FileCacheFactory
          |
          +---multiserv.util.HTMLCacheFactory
```

public class HTMLCacheFactory
extends FileCacheFactory

Constructor Index

- multiserv.util.HTMLCacheFactory()

Method Index

- getCache(String, String, long, String[], Hashtable)

Constructors

- HTMLCacheFactory

```
public HTMLCacheFactory()
```

Methods

- getCache

```
public static multiserv.util.HTMLCache getCache(String
cacheName,
                                                String dir,
                                                long
cachePeriod,
                                                String[]
suffixes,
                                                Hashtable
tokens) throws IOException
```

Class multiserv.util.HTMLDocument

```
Object
   |
   +----multiserv.util.HTMLDocument
```

public class **HTMLDocument**
extends Object

Constructor Index

- multiserv.util.HTMLDocument(String, String)
- multiserv.util.HTMLDocument(String, String, Hashtable)
- multiserv.util.HTMLDocument(String, String, Hashtable, Hashtable, boolean)

Method Index

- buildListTokens(String, String, String, Vector, Hashtable, boolean)
- buildSimpleTokens(Hashtable, Hashtable)
  Convenience method which builds a token table to be passed to sendParseDocument().
- buildTokens(Hashtable, Hashtable, boolean)
  Convenience method which builds a token table to be passed to sendParseDocument().
- toString()

Constructors

HTMLDocument

```
public HTMLDocument(String cacheName,
                    String docName) throws IOException
```

HTMLDocument

```
public HTMLDocument(String cacheName,
                    String docName,
                    Hashtable tokens) throws IOException
```

HTMLDocument

```
public HTMLDocument(String cacheName,
                    String docName,
                    Hashtable tokens,
                    Hashtable tokenData,
                    boolean checkedNameVal) throws IOException
```

Methods

buildListTokens

```
public static void buildListTokens(String cacheName,
                                   String listDocToken,
                                   String listDocName,
                                   Vector data,
                                   Hashtable tokens,
                                   boolean checkedNameVal)
throws IOException
```

- **buildSimpleTokens**

```
public static void buildSimpleTokens(Hashtable data,
                                     Hashtable tokens)
```

Convenience method which builds a token table to be passed to sendParseDocument(). It basically goes through the passed Hashtable and builds another Hashtable where the keys are renamed as ~*key*~ That is, '~*' is tacked on the beginning and '*~' is tacked on the end of each key. This method could be useful for taking the data from a database query returned in a hashtable and building a token table to populate an HTML page.

Parameters:

data - The incoming Hashtable

tokens - The token table to be built. It might already contain some tokens - which will added to.

- **buildTokens**

```
public static void buildTokens(Hashtable data,
                               Hashtable tokens,
                               boolean checkedNameVal)
```

Convenience method which builds a token table to be passed to sendParseDocument(). It basically goes through the passed Hashtable and builds another Hashtable where the keys are renamed as ~*key*~ That is, '~*' is tacked on the beginning and '*~' is tacked on the end of each key. Special cases exist for keys starting with pvt or opt. These are treated as the special token types for checkboxes/radio buttons and selection lists respectively. This method could be useful for taking the data from a database query returned in a hashtable and building a token table to populate an HTML page.

Parameters:

data - The incoming Hashtable

tokens - The token table to be built. It might already contain some tokens - which will added to.

checkedNameVal - If checked, build PVT tokens as @# name=value# @ else just as @# name# @

- **toString**

```
public java.lang.String toString()
```

Overrides:

Class multiserv.util.Logger

```
Object
   |
   +---Thread
           |
           +---multiserv.util.Logger
```

public class **Logger**
extends Thread
A class which can be used for logging.

Constructor Index

- multiserv.util.Logger(String)
  Constructor for the Logger

Method Index

- getWriter()
- main(String[])
- run()

Constructors

- **Logger**

```
public Logger(String logFile) throws IOException
```

Constructor for the Logger
Parameters:
logFile - The file to which to log

Methods

- **getWriter**

```
public static java.io.PrintWriter getWriter() throws IOException
```

- **main**

```
public static void main(String[] argv)
```

- **run**

```
public void run()
```

Overrides:
run in class Thread

Class multiserv.util.SendMail

```
Object
   |
   +---multiserv.util.SendMail
```

public class **SendMail**
extends Object
implements Runnable
This is an interim class for sending mail. Sun hava a javamail API which is still in Beta Test and will be available for platform independent sending of mail. I did download it to check it out but it required another piece of Beta software so I thought I would leave it for now.

Constructor Index

- multiserv.util.SendMail(String, String, String, String, String, String)
- multiserv.util.SendMail(String, String, String, String, String, String, boolean)

Method Index

- main(String[])
- run()

Constructors

- SendMail

```
public SendMail(String from,
                String to,
                String cc,
                String replyTo,
                String subject,
                String message) throws IOException
```

- SendMail

```
public SendMail(String from,
                String to,
                String cc,
                String replyTo,
                String subject,
                String message,
                boolean mimed) throws IOException
```

Methods

- main

```
public static void main(String[] argv)
```

- **run**

```
public void run()
```

Class multiserv.util.SoundEx

```
Object
   |
   +---multiserv.util.SoundEx
```

public class SoundEx
extends Object

Constructor Index

- multiserv.util.SoundEx()
- multiserv.util.SoundEx(String)

Method Index

- Calculate(String)
  Generates a soundex code for the input string.

Constructors

- SoundEx

```
public SoundEx()
```

- SoundEx

```
public SoundEx(String inputString)
```

Methods

- Calculate

```
public static java.lang.String Calculate(String inputString)
```

Generates a soundex code for the input string.

Parameters:

inputString - The string that is used to generate the soundex code

Returns:

A string representing the soundex code. If a code cannot be generated then the string "XXXX" is returned.

Class multiserv.util.Watcher

```
Object
   |
   +---multiserv.util.Watcher
```

public class Watcher
extends Object
implements Runnable

Constructor Index

- multiserv.util.Watcher(Watchable, long)

Method Index

- interrupt()
- run()
- start()
- stop()

Constructors

- Watcher

```
public Watcher(Watchable babe,
               long secs)
```

Methods

- interrupt

```
public void interrupt()
```

- run

```
public void run()
```

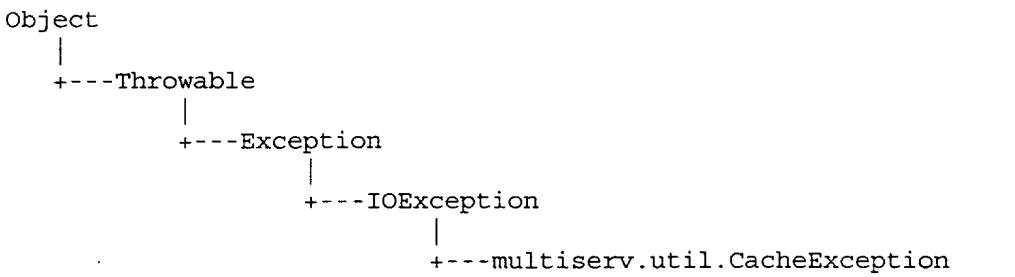
- start

```
public void start()
```

- stop

```
public void stop()
```

Class multiserv.util.CacheException



public class CacheException
extends IOException

Constructor Index

- multiserv.util.CacheException()
- multiserv.util.CacheException(String)
- multiserv.util.CacheException(String, Exception)

Method Index

- getCause()

Constructors

CacheException

```
public CacheException()
```

CacheException

```
public CacheException(String reason)
```

CacheException

```
public CacheException(String reason,
                      Exception e)
```

Methods

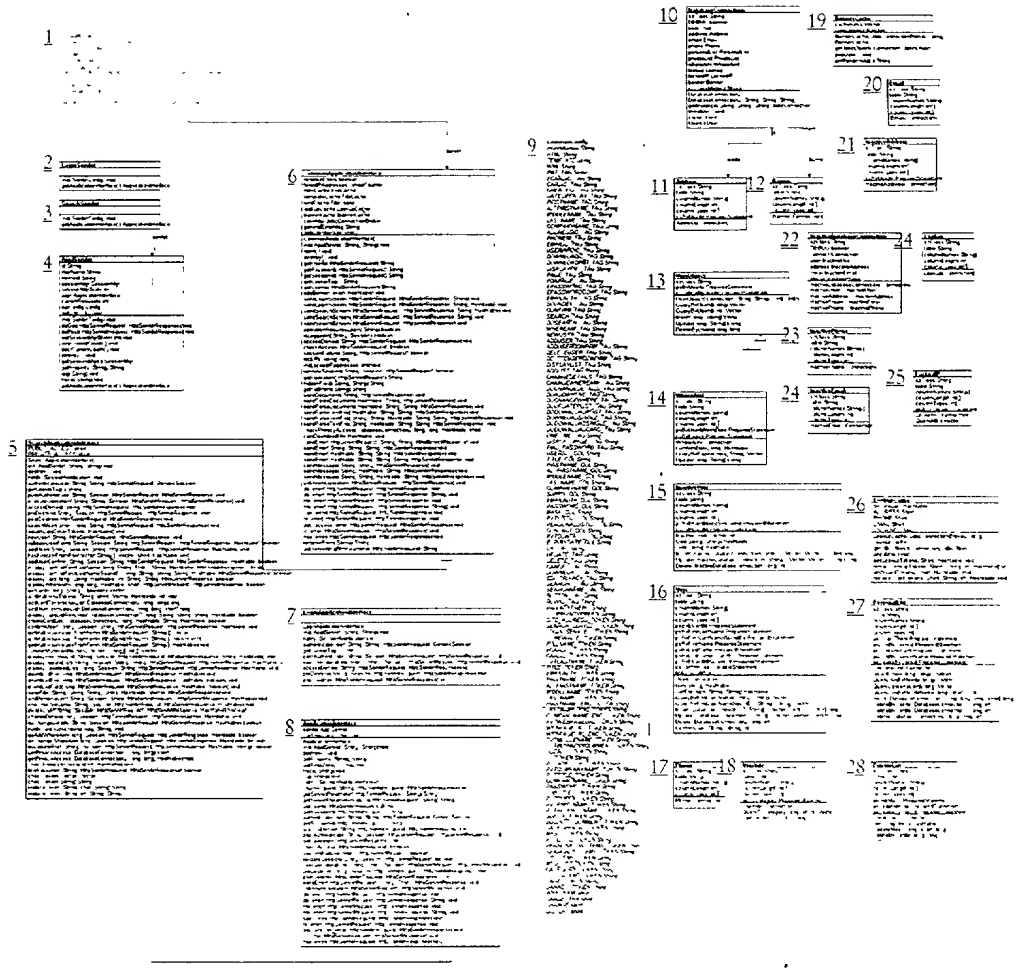
getCause

```
public java.lang.Exception getCause()
```

UML Diagrams

The following UML diagrams provide a visual description of the interactions between the Java classes used by Ecardfile.

From the high level diagram you can click on the number by the side of the class and be brought to the large scale diagram.



EcardNotifier

EcardNotifier
-parent:CommonApplicationInterface -myThread:Thread -cardId:long -eCardId:String -messageCache:String
+EcardNotifier(CommonApplicationInterface, String, long, String){constructor} +run( ).void +start( ):void +interrupt( ):void +stop( ):void +notifyUser(Hashtable, long, long, String, Hashtable):void

LoginServlet

LoginServlet
init(ServletConfig):void getApplicationInterface( ):ApplicationInterface

SearchServlet

SearchServlet
init(ServletConfig) void getApplicationInterface( ):ApplicationInterface

ApplServlet

ApplServlet
id:String mgrName:String rmiHost:String sessionMgr:SessionMgr sessionMgrState:int appl:ApplicationInterface currentRequests:int ourConfig:Config debugOn:boolean
init(ServletConfig):void doGet(HttpServletRequest, HttpServletResponse):void doPost(HttpServletRequest, HttpServletResponse):void setSessionMgrState(int):void incrCurrentCount( ):void decrCurrentCount( ):void destroy( ):void getSessionMgr( ):SessionMgr getProperty(String):String log(String):void trace(String):void getApplicationInterface( ):ApplicationInterface

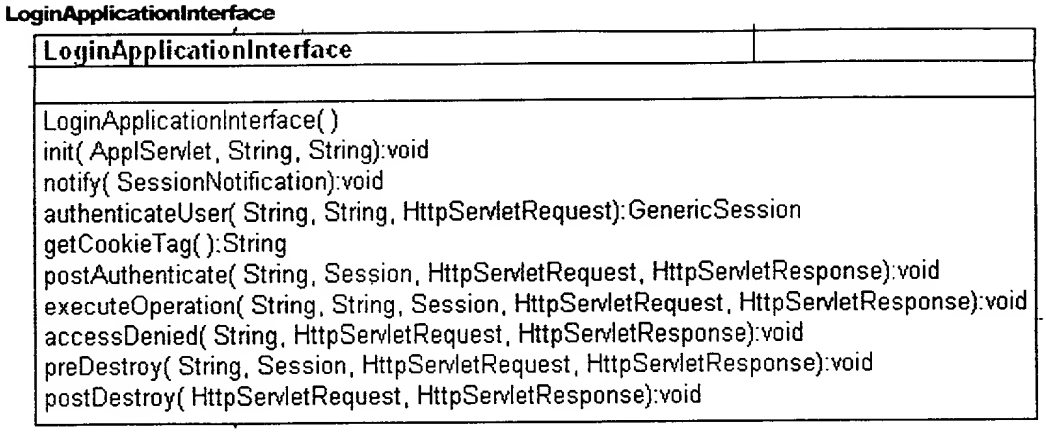
SearchApplicationInterface

SearchApplicationInterface
PUBLIC_ACCESS:short
PRIVATE_ACCESS:short
SearchApplicationInterface( )
init(ApplServlet, String, String):void
destroy( ):void
notify(SessionNotification):void
authenticateUser(String, String, HttpServletRequest):GenericSession
getCookieTag( ):String
postAuthenticate(String, Session, HttpServletRequest, HttpServletResponse):void
executeOperation(String, String, Session, HttpServletRequest, HttpServletResponse).void
accessDenied(String, HttpServletRequest, HttpServletResponse):void
preDestroy(String, Session, HttpServletRequest, HttpServletResponse):void
postDestroy(HttpServletRequest, HttpServletResponse):void
searchUser(String, long, String, HttpServletRequest, HttpServletResponse):void
adjustCardDetailTokens(Hashtable):void
newUser(String, HttpServletRequest, HttpServletResponse):void
validateUserForm(String, Session, String, HttpServletRequest, HttpServletResponse, Hashtable)
addUser(String, Session, String, HttpServletRequest, HttpServletResponse, Hashtable) void
hashVectorFromRowVector(String[], Vector, short, Hashtable):void
addUserConfirm(String, Session, String, HttpServletRequest, HttpServletResponse, Hashtable).bo
displayCardListFirstLastName(long, String, String, String, Hashtable, HttpServletResponse):boole
displayCardListFirstLastNameSoundEx(long, String, String, String, Hashtable, HttpServletRespon
displayCard(long, String, Hashtable, int, String, String, HttpServletResponse):boolean
displayWhereAml(long, long, Hashtable, short, HttpServletRequest, HttpServletResponse).boolea
getCards(long, String[], boolean):Vector
addMultiRowTokens(String, short, Vector, Hashtable, int):void
addCardToPersonalList(DatabaseConnection2, long, long):long
addUserToPrivateList(DatabaseConnection2, long, long, short):long
displayCardListPersonal(DatabaseConnection2, long, String, String, String, Hashtable).boolean
createCardList(DatabaseConnection2, long, Hashtable, String, Hashtable):boolean
confirmUser(String, Session, String, HttpServletRequest, HttpServletResponse, Hashtable) void
getMultiRowValuesFromForm(HttpServletRequest, String[]) Vector
getMultiRowValuesFromForm(HttpServletRequest, String[], String) Vector
getMultiRowValuesFromForm(HttpServletRequest, String[], Hashtable) void
convertMultiRowHashes(Vector, String[], int[]).Vector
displayPersonalList(String, Session, HttpServletRequest, HttpServletResponse, String, Hashtable
displayUpdateList(String, Session, String, String, HttpServletRequest, HttpServletResponse, Has
displayDownloadList(String, Session, String, HttpServletRequest, HttpServletResponse, Hashtabl
downloadFile(long, HttpServletRequest, HttpServletResponse, Hashtable):void
downloadFile(long, HttpServletRequest, HttpServletResponse, Hashtable, boolean):void
downloadCard(long, HttpServletRequest, HttpServletResponse, Hashtable, boolean):void
sendFile(String, String, String, String, Hashtable, Vector, HttpServletResponse).void
addPersonalList(String, Session, String, HttpServletRequest, HttpServletResponse, Hashtable) vo
oneTimeWelcome(String, Session, HttpServletRequest, HttpServletResponse, Hashtable):void
deleteUser(String, Session, HttpServletRequest, HttpServletResponse, Hashtable):boolean
changeDetails(long, Session, HttpServletRequest, HttpServletResponse, Hashtable):void
doChangeDetails(String, Session, HttpServletRequest, HttpServletResponse, Hashtable):boolean
notifyCardSubscribers(long, String):void
doAddWhereAml(long, Session, HttpServletRequest, HttpServletResponse, Hashtable):boolean
doChangeWhereAml(long, Session, HttpServletRequest, HttpServletResponse, Hashtable):boolea
doUpdatePlist(String, Session, HttpServletRequest, HttpServletResponse, Hashtable, String) boo
getPrivacyAccess(DatabaseConnection2, long, long):short
getPrivacyAccess(DatabaseConnection2, long, long, Hashtable).void
checkPrivacyAccess(short, Hashtable) void
findPassword(String, HttpServletRequest, HttpServletResponse) boolean
checkToken(Vector) Vector
checkToken(String).String
replaceToken(String, char, String) String
replaceToken(String, int, String):String

CommonApplicationInterface

CommonApplicationInterface
verboseErrors boolean failedIPAddresses TimedCounter htmlCache.FileCache templateCache.FileCache wmlCache.FileCache lookupCache.LookupCache bannerCache.BannerCache connMgr.JdbcConnectionBroker2 generalErrorMsg.String defaultPdaPage String
CommonApplicationInterface( ) init(ApplServlet, String, String) void reinit( ) void destroy( ) void getUserId(HttpServletRequest) String getPassword(HttpServletRequest) String getSessionId(HttpServletRequest) String getCookieTag( ) String getCookie(HttpServletRequest) String addBannerToken(Hashtable) void sendLoginScreen(HttpServletRequest, HttpServletResponse, String) void sendLoginScreen(HttpServletRequest, HttpServletResponse, String, Hashtable) void sendSearchScreen(HttpServletRequest, HttpServletResponse, String, Hashtable) void sendSearchScreen(HttpServletRequest, HttpServletResponse, String) void sendSearchScreen(HttpServletRequest, HttpServletResponse) void operationRequiresLogin(String) boolean isLoggedIn(String, Session) boolean accessDenied(String, HttpServletRequest, HttpServletResponse) void checkAccess(HttpServletRequest) boolean sessionFailure(String, HttpServletRequest) boolean lockIP(String) long initLockedIPaddresses(int) void validateSession(String, Session, HttpServletRequest) boolean getOperation(HttpServletRequest) String hiddenField(String, String) String getFullPath(String) String sendDocument(String, HttpServletResponse) void sendParseDocument(Hashtable, String, HttpServletResponse) void sendParseDocument(Hashtable, String, String, HttpServletResponse) void sendParseTextFile(Hashtable, String, String, HttpServletResponse) void sendParseTextFile(String, String, Hashtable, String, String, HttpServletResponse) void sendParseTextFile(String, Hashtable, String, String, HttpServletResponse) void checkPrivacyAccess(DatabaseConnection2, long, long, Hashtable) short cardDownloadUrl(Hashtable) void sendError(HttpServletRequest, String, String, HttpServletResponse) void sendError(String, String, String, HttpServletResponse) void sendError(Hashtable, String, String, HttpServletResponse) void sendError(Hashtable, String, String, String, HttpServletResponse) void sendMessage(String, String, HttpServletResponse) void sendMessage(String, Hashtable, String, HttpServletResponse) void sendMessage(String, String, Hashtable, String, HttpServletResponse) void unknownOperation(HttpServletRequest, HttpServletResponse) void db_error(HttpServletRequest, HttpServletResponse) void db_error(HttpServletRequest, HttpServletResponse, String) void nfe_error(HttpServletRequest, HttpServletResponse) void nfe_error(HttpServletRequest, HttpServletResponse, String) void sae_error(HttpServletRequest, HttpServletResponse) void re_error(HttpServletRequest, HttpServletResponse) void doc_access_error(HttpServletRequest, HttpServletResponse) void io_error(HttpServletRequest, HttpServletResponse) void nse_error(HttpServletRequest, HttpServletResponse) void verboseError(String) String getServletImgBtnParameter(HttpServletRequest) String

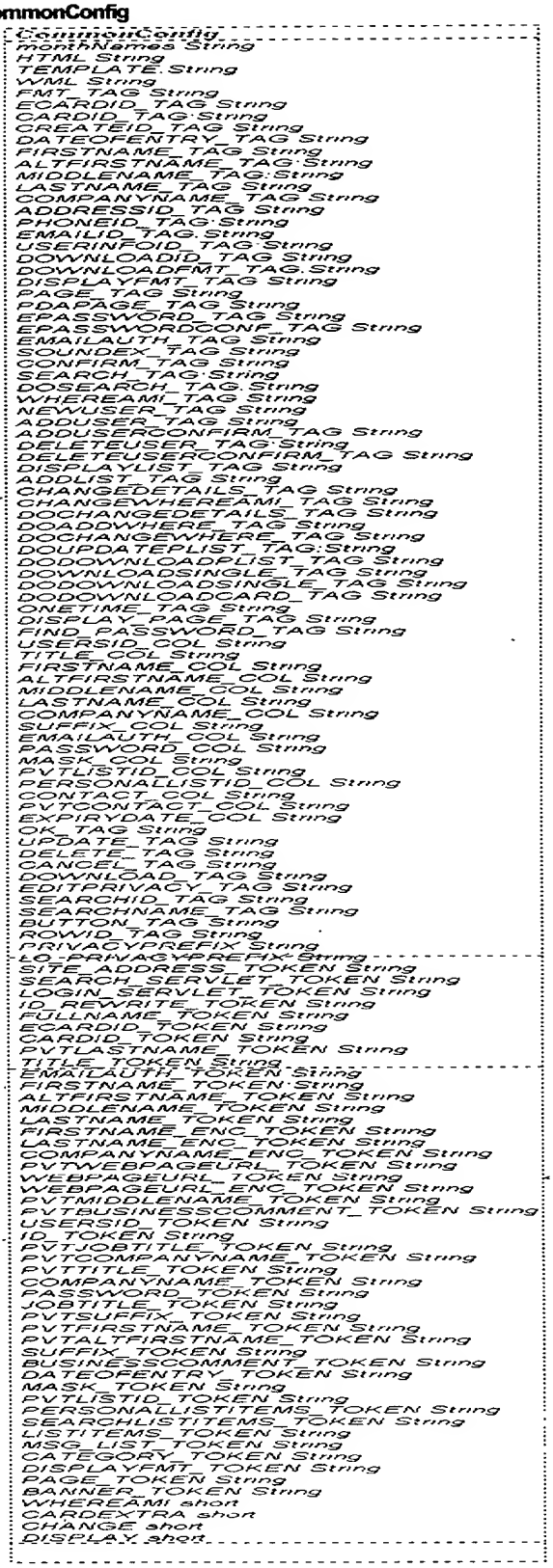
LoginApplicationInterface



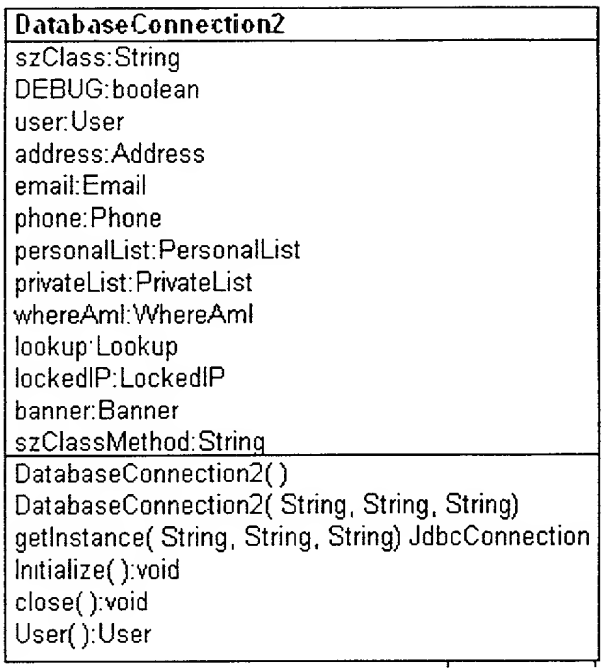
ApplicationInterface

ApplicationInterface
servlet:ApplServlet ourTimeZone:TimeZone
ApplicationInterface( ) init(ApplServlet, String, String):void destroy( ):void getProperty(String):String getTimeZone( ):TimeZone trace(String):void log(String):void notify(SessionNotification):void chainRequest(String, HttpServletRequest, HttpServletResponse):void getServletParameter(HttpServletRequest, String):String getServletParameterValues(HttpServletRequest, String):String[] getUserId(HttpServletRequest):String getPassword(HttpServletRequest):String authenticateUser(String, String, HttpServletRequest):GenericSession getSessionId(HttpServletRequest):String accessDenied(String, HttpServletRequest, HttpServletResponse).void postAuthenticate(String, Session, HttpServletRequest, HttpServletResponse):void getOperation(HttpServletRequest):String checkAccess(HttpServletRequest):boolean sessionFailure(String, HttpServletRequest):boolean validateSession(String, Session, HttpServletRequest).boolean executeOperation(String, String, Session, HttpServletRequest, HttpServletResponse):void preDestroy(String, Session, HttpServletRequest, HttpServletResponse) void postDestroy(HttpServletRequest, HttpServletResponse):void sendError(HttpServletRequest, String, String, HttpServletResponse) void unknownOperation(HttpServletRequest, HttpServletResponse):void db_error(HttpServletRequest, HttpServletResponse):void db_error(HttpServletRequest, HttpServletResponse, String) void nfe_error(HttpServletRequest, HttpServletResponse):void nfe_error(HttpServletRequest, HttpServletResponse, String) void sae_error(HttpServletRequest, HttpServletResponse).void re_error(HttpServletRequest, HttpServletResponse) void doc_access_error(HttpServletRequest, HttpServletResponse):void io_error(HttpServletRequest, HttpServletResponse):void nse_error(HttpServletRequest, HttpServletResponse):void

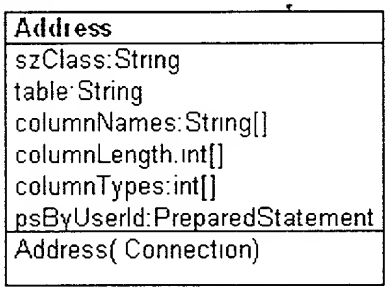
CommonConfig



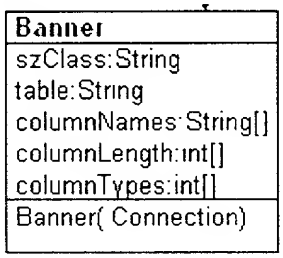
DatabaseConnection2



Address



Banner



UserObject

UserObject
szClass:String psByUserId:PreparedStatement psDeleteByUserId:PreparedStatement
UserObject(Connection, String, String[], int[], int[]) QueryByUserId(long):Vector QueryByUserId(int):Vector Insert(long, String[]):long Update(long, String[]):long DeleteByUserId(long):long

WhereAml

WhereAml
szClass:String table:String columnNames:String[] columnLength:int[] columnTypes:int[] psByUserIdWithDate:PreparedStatement psByExpiry:PreparedStatement
WhereAml(Connection) GetWithDate(long):Hashtable QueryByExpiry(long, String):Vector Update(long, String[]):long

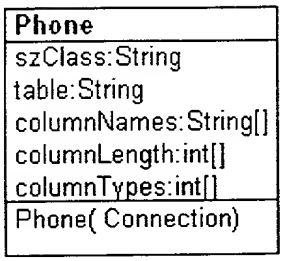
InactiveUser

InactiveUser
szClass:String table:String columnNames:String[] columnLength:int[] columnTypes:int[] psByECardIdAndSessionId:PreparedStatement psByECardId:PreparedStatement
InactiveUser(Connection) Get(String, String):Hashtable Get(String):Hashtable Insert(InactiveDatabaseConnection, String[], Vector, Vector, Vector):long Update(InactiveDatabaseConnection, String[], Vector, Vector, Vector):long Delete(InactiveDatabaseConnection, long):int

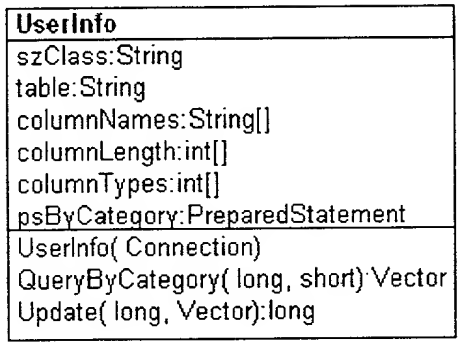
User

User
szClass:String table:String columnNames:String[] columnLength:int[] columnTypes:int[] psByECardId:PreparedStatement psByFirstLastName:PreparedStatement psByFirstLastNameSoundEx:PreparedStatement psByFirstName.PreparedStatement psByLastName.PreparedStatement psByLastNameSoundEx:PreparedStatement psByECardIdPassword:PreparedStatement csConfirmUser:CallableStatement MAX_ROWS:int
User(Connection) Get(String):Hashtable GetForLogin(String, String):Hashtable QueryByFirstLastName(String, String):Vector QueryByFirstLastNameSoundEx(String, String):Vector Insert(DatabaseConnection2, String[], Vector, Vector, Vector):long Update(DatabaseConnection2, String[], Vector, Vector, Vector) long Delete(DatabaseConnection2, long):int ConfirmUser(String, String):int

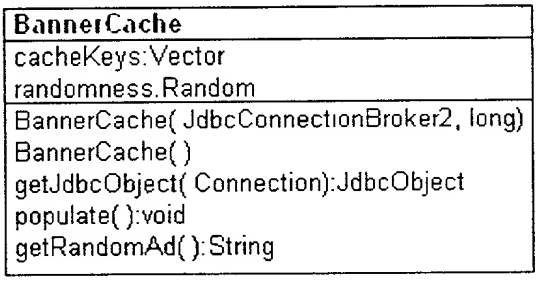
Phone



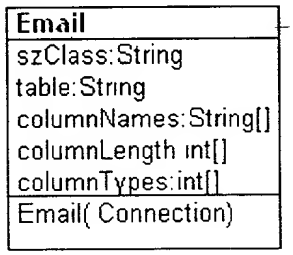
UserInfo



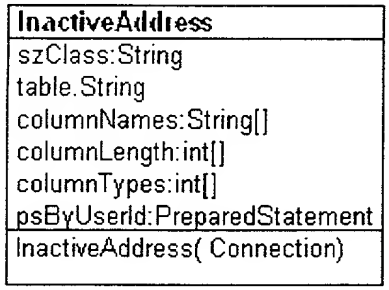
BannerCache



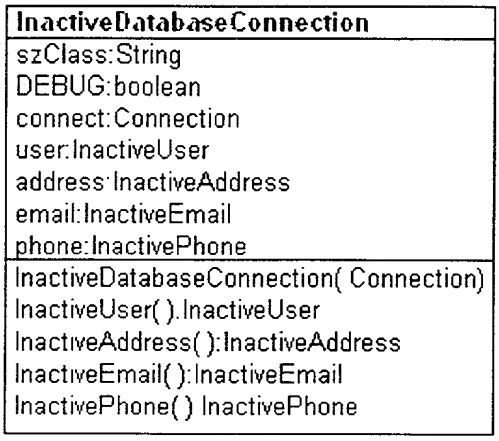
EMail



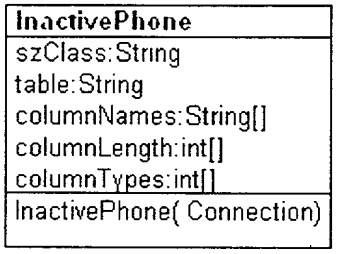
InactiveAddress



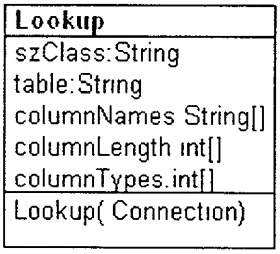
InactiveDatabaseConnection



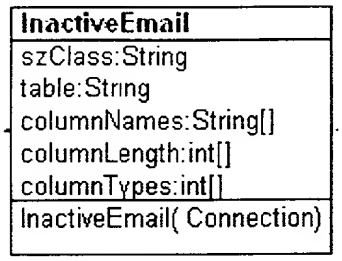
InactivePhone



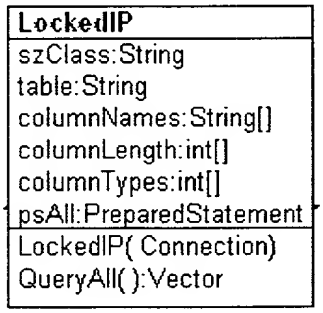
Lookup



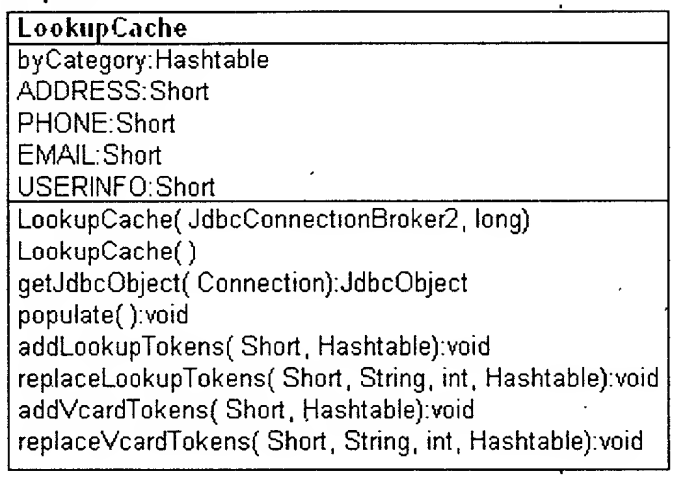
InactiveEmail



LockedIP



LookupCache

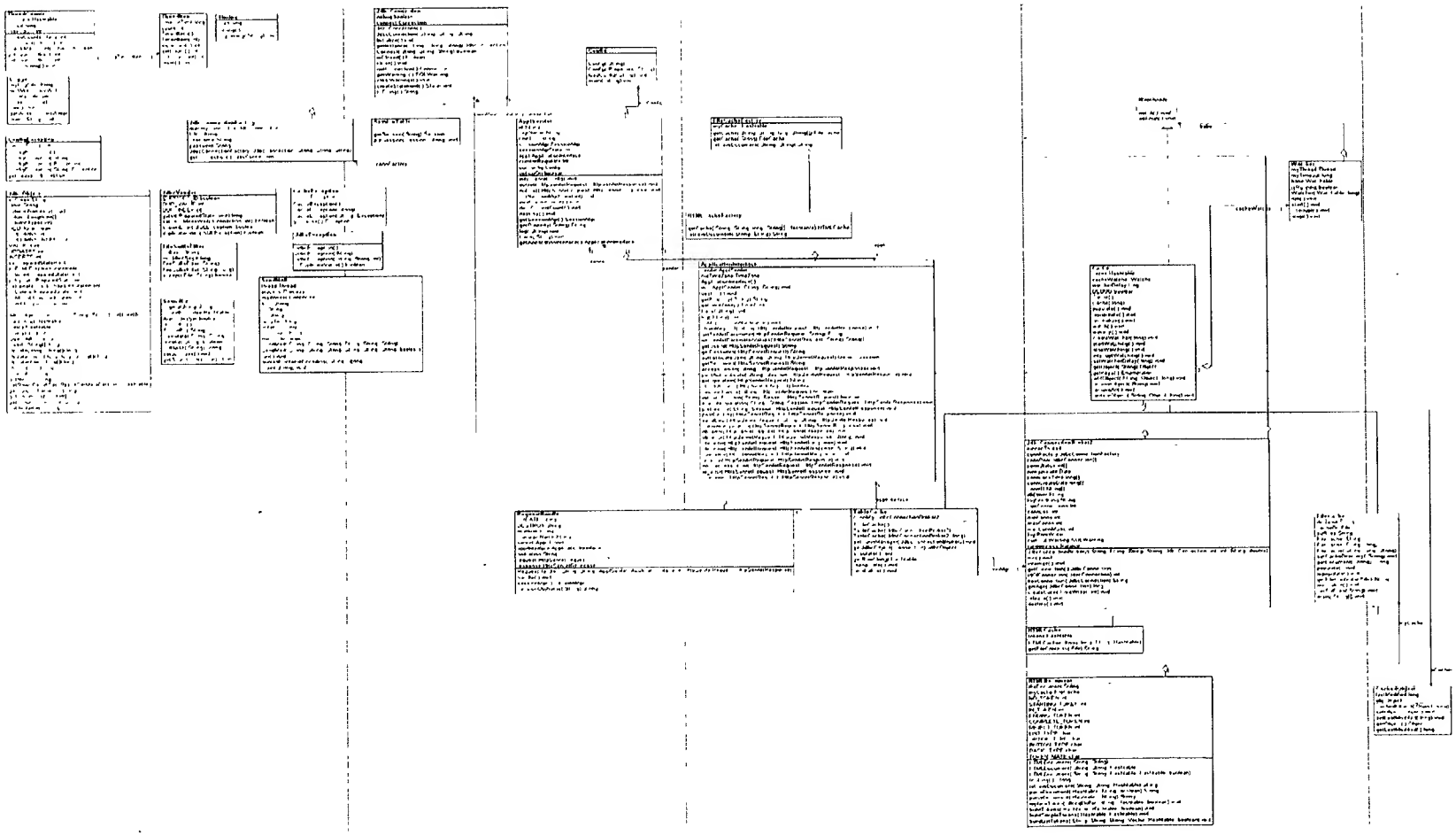


PersonalList

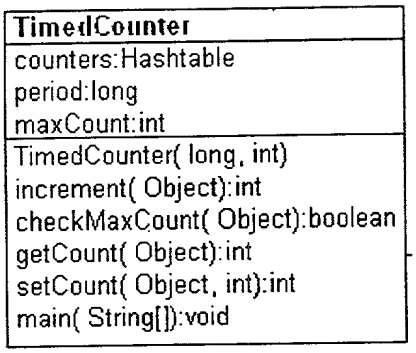
PersonalList
szClass:String table:String columnNames:String[] columnLength:int[] columnTypes:int[] psIsCardThere:PreparedStatement psJoinByUsersId:PreparedStatement psContainsCard:PreparedStatement psJoinByUsersIdName:PreparedStatement psDeleteByCardId:PreparedStatement
PersonalList(Connection) IsCardThere(long, long):boolean QueryJoinByUserId(long):Vector QueryContainsCard(long):Vector QueryJoinByUserIdName(long, char):Vector Insert(DatabaseConnection2, long, long, long, short):long DeleteByUserId(DatabaseConnection2, long):int DeleteByCardId(DatabaseConnection2, long) int Delete(DatabaseConnection2, long, long):int

PrivateList

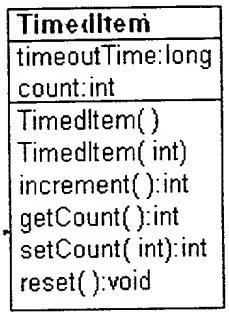
PrivateList
szClass:String table.String columnNames:String[] columnLength:int[] columnTypes:int[] psGetMask:PreparedStatement psUpdateMask:PreparedStatement psDeleteByCardId:PreparedStatement
PrivateList(Connection) Get(long, long):Hashtable UpdateMask(long, short) long DeleteByCardId(long):long



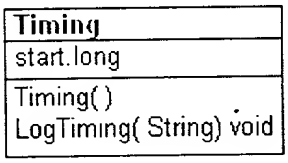
TimedCounter



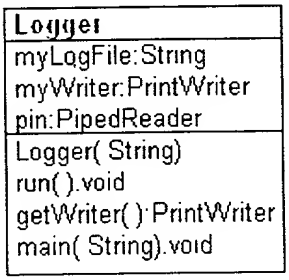
TimedItem



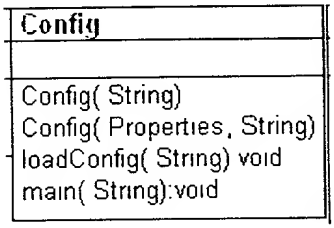
Timing



Logger



Config



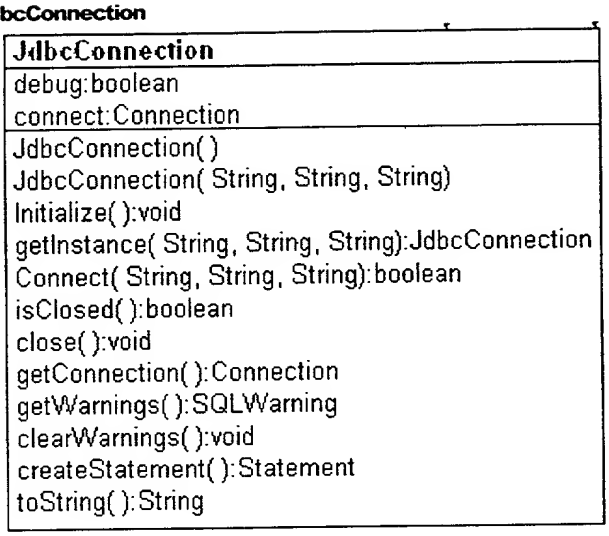
ConfigException

ConfigException
cause:Exception
ConfigException( ) ConfigException(String) ConfigException(Exception) ConfigException(String, Exception) getCause( ):Exception

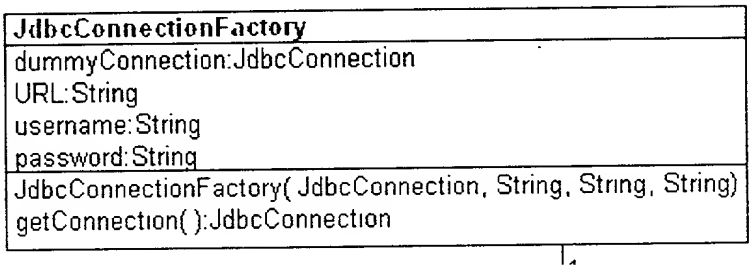
JdbcObject

JdbcObject
szClass:String table:String columnNames:String[] columnLength:int[] columnTypes.int[] DEBUG:boolean COLUMNS:int COLUMNS_INSERT:int VALUES.int UPDATES:int INSERTS:int ps:PreparedStatement psById:PreparedStatement psInsert:PreparedStatement psUpdate:PreparedStatement psUpdateCheck:PreparedStatement psDelete:PreparedStatement psMaxId:PreparedStatement psAll:PreparedStatement connect:Connection
JdbcObject(Connection, String, String[], int[], int[]) Get(long):Hashtable Get( ):Hashtable Query( ):Vector QueryAll( ):Vector Insert(String[]):long Update(long, String[]):long UpdateCheck(long, long, String[]):long UpdateRow(String[]):long Delete(long) long Execute( ).long getNextId( ):long getRow(ResultSet, ResultSetMetaData, int):Hashtable getColumnNames( ):String[] getColumnTypes( ) int[] getColumnString(int).String TableName( ):String

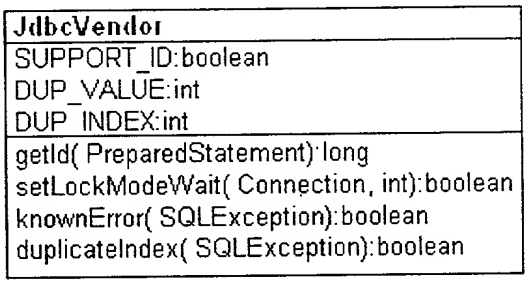
JdbcConnection



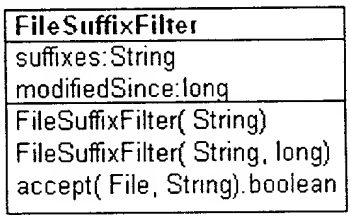
JdbcConnectionFactory



JdbcVendor



FileSuffixFilter



SoundEx

SoundEx
originalString:String soundExCodes:Hashtable AreCodesSet:boolean
SoundEx( ) SoundEx(String) Calculate(String):String isValid(String):boolean collapse(String):String setupCodes( ):void getSoundCode(char):short

CacheException

CacheException
cause:Exception
CacheException( ) CacheException(String) CacheException(String, Exception) getCause( ):Exception

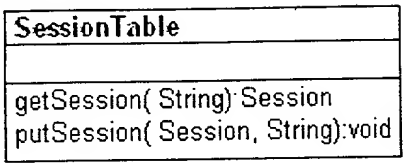
JdbcException

JdbcException
JdbcException( ) JdbcException(String) JdbcException(String, String, int) isDuplicateValue( ):boolean

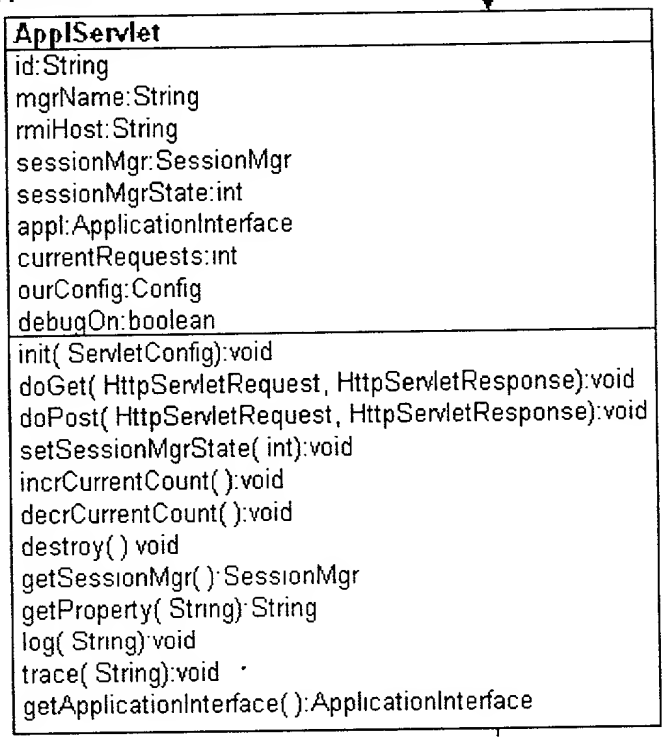
SendMail

SendMail
thread.Thread process:Process mailWriter:PrintWriter from:String to.String cc.String replyTo.String subject:String message:String mimed:boolean
SendMail(String, String, String, String, String, String) SendMail(String, String, String, String, String, String, boolean) run( ) void quotedPrintableEncoding(String) String main(String):void

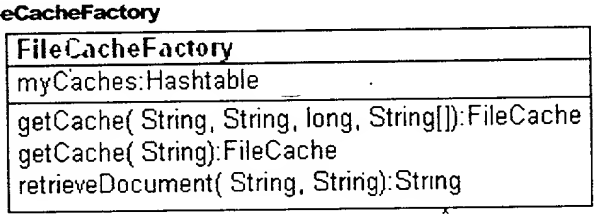
SessionTable



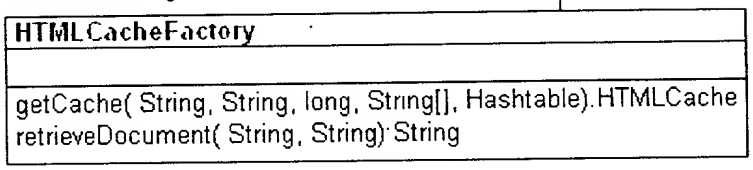
ApplServlet



FileCacheFactory



HTMLCacheFactory



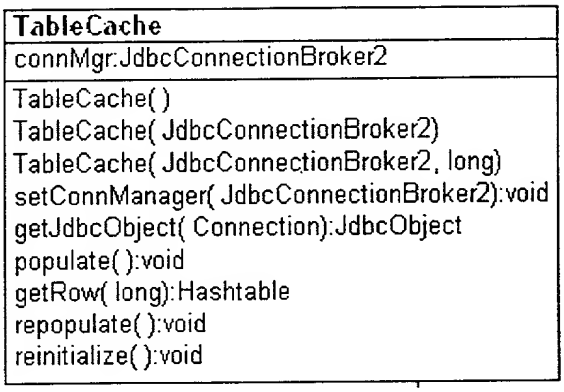
RequestHandler

RequestHandler
CREATE String DESTROY.String rmiHost:String managerName String servlet.ApplServlet applInterface ApplicationInterface operation.String request HttpServletRequest response:HttpServletResponse
RequestHandler(String, String, ApplServlet, ApplicationInterface, HttpServletRequest, HttpServletResponse) handle( ) void sessionMgr( ).SessionMgr sessionObjName(String).String

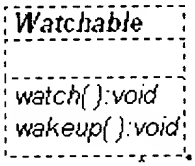
ApplicationInterface

ApplicationInterface
servlet:ApplServlet ourTimeZone:TimeZone
ApplicationInterface( ) init(ApplServlet, String, String):void destroy( ):void getProperty(String):String getTimeZone( ):TimeZone trace(String):void log(String):void notify(SessionNotification):void chainRequest(String, HttpServletRequest, HttpServletResponse):void getServletParameter(HttpServletRequest, String):String getServletParameterValues(HttpServletRequest, String):String[] getUserId(HttpServletRequest):String getPassword(HttpServletRequest):String authenticateUser(String, String, HttpServletRequest):GenericSession getSessionId(HttpServletRequest):String accessDenied(String, HttpServletRequest, HttpServletResponse):void postAuthenticate(String, Session, HttpServletRequest, HttpServletResponse):void getOperation(HttpServletRequest):String checkAccess(HttpServletRequest):boolean sessionFailure(String, HttpServletRequest):boolean validateSession(String, Session, HttpServletRequest):boolean executeOperation(String, String, Session, HttpServletRequest, HttpServletResponse):void preDestroy(String, Session, HttpServletRequest, HttpServletResponse):void postDestroy(HttpServletRequest, HttpServletResponse):void sendError(HttpServletRequest, String, String, HttpServletResponse):void unknownOperation(HttpServletRequest, HttpServletResponse):void db_error(HttpServletRequest, HttpServletResponse):void db_error(HttpServletRequest, HttpServletResponse, String):void nfe_error(HttpServletRequest, HttpServletResponse):void nfe_error(HttpServletRequest, HttpServletResponse, String):void sae_error(HttpServletRequest, HttpServletResponse):void re_error(HttpServletRequest, HttpServletResponse):void doc_access_error(HttpServletRequest, HttpServletResponse):void io_error(HttpServletRequest, HttpServletResponse):void nse_error(HttpServletRequest, HttpServletResponse):void

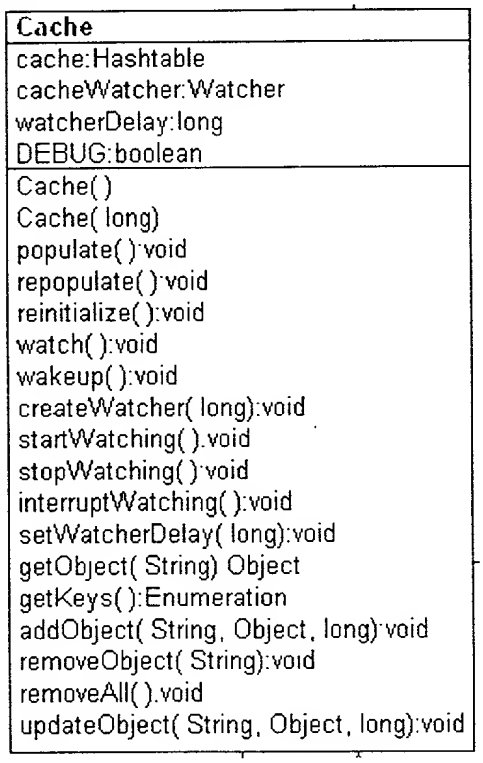
TableCache



Watchable



Cache



JdbcConnectionBroker2

JdbcConnectionBroker2
runner:Thread connFactory:JdbcConnectionFactory connPool.JdbcConnection[] connStatus:int[] now:java.util.Date connLockTime:long[] connCreateDate:long[] connID:String[] dbDriver:String logFileString:String currConnections:int connLast.int minConns.int maxConns.int maxConnMSec:int log:PrintWriter currSQLWarning:SQLWarning randomness:Random
JdbcConnectionBroker2(String, String, String, String, JdbcConnection, int, int, String, double) run( ):void interrupt( ):void getConnection( ):JdbcConnection idOfConnection(JdbcConnection):int freeConnection(JdbcConnection):String getAge(JdbcConnection):long createConn(PrintWriter, int) void release( ) void destroy( ):void

HTMLCache

HTMLCache
tokens:Hashtable
HTMLCache(String, long, String, Hashtable) getFileContents(File):String

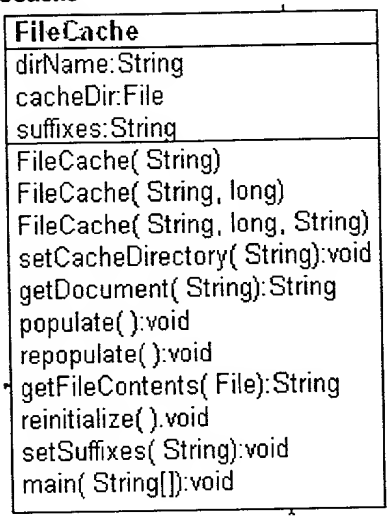
HTMLDocument

HTMLDocument
theDocument:String myCache:FileCache NO_TOKEN:int STARTING_TOKEN:int IN_TOKEN:int ENDING_TOKEN:int COMPLETE_TOKEN:int REJECT_TOKEN:int LIST_TYPE:char OPTION_TYPE:char BUTTON_TYPE:char BASIC_TYPE:char TOKEN_MATE:char
HTMLDocument(String, String) HTMLDocument(String, String, Hashtable) HTMLDocument(String, String, Hashtable, Hashtable, boolean) toString( ):String retrieveDocument(String, String, Hashtable):String parseDocument(Hashtable, String, boolean):String parseDocument(Hashtable, String):String replaceToken(StringBuffer, String, Hashtable, boolean):void buildTokens(Hashtable, Hashtable, boolean):void buildSimpleTokens(Hashtable, Hashtable):void buildListTokens(String, String, String, Vector, Hashtable, boolean):void

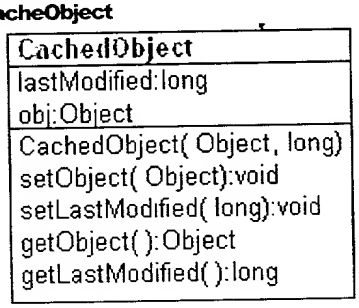
Watcher

Watcher
myThread.Thread myTimeout:long babe:Watchable isRunning boolean
Watcher(Watchable, long) run( ):void start( ):void interrupt( ):void stop( ):void

FileCache

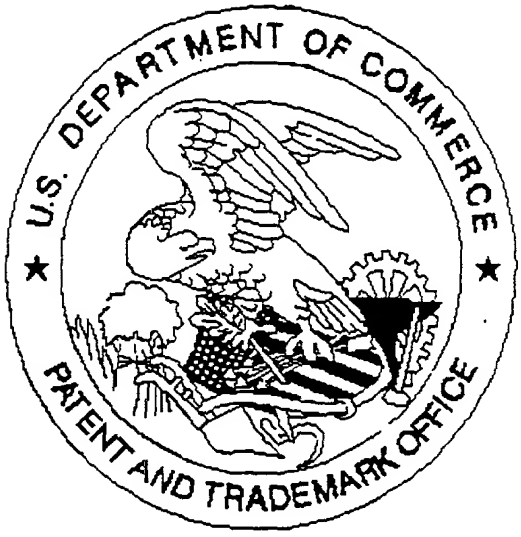


CacheObject



United States Patent & Trademark Office

Office of Initial Patent Examination -- Scanning Division



Application deficiencies were found during scanning:

☐ Page(s)_____ of Transmittal were not present for scanning. (Document title)

☐ Page(s)_____ of _____ were not present for scanning. (Document title)

☐ Scanned copy is best available.